When ‘false’ models predict better than ‘true’ ones: Paradoxes of the bias-variance tradeoff

Hemank Lamba <hlamba@cs.cmu.edu>
Momin M. Malik <momin.malik@cs.cmu.edu>

10-702/36-702: Statistical Machine Learning
Thursday, May 4, 2017

Prediction vs. explanation

- Classical statistics focused on inference and unbiased estimators
- Why? Wanted *explanations* of or *information* about underlying data-generating processes
- ML focuses on *prediction*. A separate goal (Shmueli, 2010; Breiman, 2001), often with very different methods
- But... why should prediction be different? How can a model that predicts better be further from the 'truth'?

Prediction vs. explanation

- Classical statistics focused on inference and unbiased estimators
- Why? Wanted *explanations* of or *information* about underlying data-generating processes
- ML focuses on *prediction*. A separate goal (Shmueli, 2010; Breiman, 2001), often with very different methods
- But... why should prediction be different? How can a model that predicts better be further from the 'truth'?

Prediction vs. explanation

- Classical statistics focused on inference and unbiased estimators
- Why? Wanted *explanations* of or *information* about underlying data-generating processes
- ML focuses on *prediction*. A separate goal (Shmueli, 2010; Breiman, 2001), often with very different methods
- But... why should prediction be different? How can a model that predicts better be further from the 'truth'?

Prediction vs. explanation

- Classical statistics focused on inference and unbiased estimators
- Why? Wanted *explanations* of or *information* about underlying data-generating processes
- ML focuses on *prediction*. A separate goal (Shmueli, 2010; Breiman, 2001), often with very different methods
- But... why should prediction be different? How can a model that predicts better be further from the 'truth'?

Prediction vs. explanation

- Classical statistics focused on inference and unbiased estimators
- Why? Wanted *explanations* of or *information* about underlying data-generating processes
- ML focuses on *prediction*. A separate goal (Shmueli, 2010; Breiman, 2001), often with very different methods
- But... why should prediction be different? How can a model that predicts better be further from the 'truth'?

Prediction vs. explanation

- Classical statistics focused on inference and unbiased estimators
- Why? Wanted *explanations* of or *information* about underlying data-generating processes
- ML focuses on *prediction*. A separate goal (Shmueli, 2010; Breiman, 2001), often with very different methods
- But... why should prediction be different? How can a model that predicts better be further from the 'truth'?

Prediction vs. explanation

- Classical statistics focused on inference and unbiased estimators
- Why? Wanted *explanations* of or *information* about underlying data-generating processes
- ML focuses on *prediction*. A separate goal (Shmueli, 2010; Breiman, 2001), often with very different methods
- But... why should prediction be different? How can a model that predicts better be further from the 'truth'?

A ‘false’ model that predicts better than a ‘true’ one

- All models are ‘wrong’ (Box, 1979), usually because we haven’t or can’t measure all causal variables or we have the wrong function class
- If we have all causal variables and the right function class, then a ‘false’ model is one with the wrong variables (“misspecified”) (and/or with incorrect estimates/inferences)
- Can a ‘wrong’ model predict better? Yes.

A ‘false’ model that predicts better than a ‘true’ one

- All models are ‘wrong’ (Box, 1979), usually because we haven’t or can’t measure all causal variables or we have the wrong function class
- If we have all causal variables and the right function class, then a ‘false’ model is one with the wrong variables (“misspecified”) (and/or with incorrect estimates/inferences)
- Can a ‘wrong’ model predict better? Yes.

A ‘false’ model that predicts better than a ‘true’ one

- All models are ‘wrong’ (Box, 1979), usually because we haven’t or can’t measure all causal variables or we have the wrong function class
- If we have all causal variables and the right function class, then a ‘false’ model is one with the wrong variables (“misspecified”) (and/or with incorrect estimates/inferences)
- Can a ‘wrong’ model predict better? Yes.

A ‘false’ model that predicts better than a ‘true’ one

- All models are ‘wrong’ (Box, 1979), usually because we haven’t or can’t measure all causal variables or we have the wrong function class
- If we have all causal variables and the right function class, then a ‘false’ model is one with the wrong variables (“misspecified”) (and/or with incorrect estimates/inferences)
- Can a ‘wrong’ model predict better? Yes.

A ‘false’ model that predicts better than a ‘true’ one

- All models are ‘wrong’ (Box, 1979), usually because we haven’t or can’t measure all causal variables or we have the wrong function class
- If we have all causal variables and the right function class, then a ‘false’ model is one with the wrong variables (“misspecified”) (and/or with incorrect estimates/inferences)
- Can a ‘wrong’ model predict better? Yes.

A ‘false’ model that predicts better than a ‘true’ one

- All models are ‘wrong’ (Box, 1979), usually because we haven’t or can’t measure all causal variables or we have the wrong function class
- If we have all causal variables and the right function class, then a ‘false’ model is one with the wrong variables (“misspecified”) (and/or with incorrect estimates/inferences)
- Can a ‘wrong’ model predict better? Yes.

A ‘false’ model that predicts better than a ‘true’ one

- All models are ‘wrong’ (Box, 1979), usually because we haven’t or can’t measure all causal variables or we have the wrong function class
- If we have all causal variables and the right function class, then a ‘false’ model is one with the wrong variables (“misspecified”) (and/or with incorrect estimates/inferences)
- Can a ‘wrong’ model predict better? Yes.

A 'false' model that predicts better than a 'true' one

Setting:

- True (generative) model: $\mathbf{y} = \beta_p \mathbf{X}_p + \beta_q \mathbf{X}_q + \varepsilon$ with $\varepsilon \overset{iid}{\sim} N(\mathbf{0}, \sigma^2 \mathbf{I})$
- Underspecified model: $\mathbf{y} = \beta_p \mathbf{X}_p + \varepsilon$

Theoretical result (Wu, Harris, and McAuley, 2007; Shmueli, 2010):

- An underspecified fit has lower expected MSE when, for $\mathbf{H}_p = \mathbf{X}_p(\mathbf{X}_p^T \mathbf{X}_p)^{-1} \mathbf{X}_p^T$,

$$R_c := \beta_q^T \mathbf{X}_q^T (\mathbf{I}_n - \mathbf{H}_p) \mathbf{X}_q \beta_q < q\sigma^2$$

  E.g., when $\|\beta_q\|_1$ is small, σ^2 is large, or β_p and β_q are correlated.

- To illustrate, take $p = 10$, $q = 3$, $\beta_p = 10 \cdot \mathbf{1}_p$, $\beta_q = \mathbf{1}_q$, and $\sigma = 10 \cdot \sqrt{R_c}$.

A 'false' model that predicts better than a 'true' one

Setting:

- True (generative) model: $\mathbf{y} = \beta_p \mathbf{X}_p + \beta_q \mathbf{X}_q + \varepsilon$ with $\varepsilon \overset{iid}{\sim} N(\mathbf{0}, \sigma^2 \mathbf{I})$
- Underspecified model: $\mathbf{y} = \beta_p \mathbf{X}_p + \varepsilon$

Theoretical result (Wu, Harris, and McAuley, 2007; Shmueli, 2010):

- An underspecified fit has lower expected MSE when, for $\mathbf{H}_p = \mathbf{X}_p(\mathbf{X}_p^T \mathbf{X}_p)^{-1} \mathbf{X}_p^T$,

$$R_\varepsilon := \beta_q^T \mathbf{X}_q^T (\mathbf{I}_n - \mathbf{H}_p) \mathbf{X}_q \beta_q < q\sigma^2$$

  E.g., when $\|\beta_q\|_1$ is small, σ^2 is large, or β_p and β_q are correlated.

- To illustrate, take $p = 10$, $q = 3$, $\beta_p = 10 \cdot \mathbf{1}_p$, $\beta_q = \mathbf{1}_q$, and $\sigma = 10 \cdot \sqrt{R_\varepsilon}$.

A 'false' model that predicts better than a 'true' one

Setting:

- True (generative) model: $\mathbf{y} = \beta_p \mathbf{X}_p + \beta_q \mathbf{X}_q + \varepsilon$ with $\varepsilon \overset{iid}{\sim} N(\mathbf{0}, \sigma^2 \mathbf{I})$
- Underspecified model: $\mathbf{y} = \beta_p \mathbf{X}_p + \varepsilon$

Theoretical result (Wu, Harris, and McAuley, 2007; Shmueli, 2010):

- An underspecified fit has lower expected MSE when, for $\mathbf{H}_p = \mathbf{X}_p(\mathbf{X}_p^T \mathbf{X}_p)^{-1} \mathbf{X}_p^T$,

$$R_c := \beta_q^T \mathbf{X}_q^T (\mathbf{I}_n - \mathbf{H}_p) \mathbf{X}_q \beta_q < q\sigma^2$$

E.g., when $\|\beta_q\|_1$ is small, σ^2 is large, or β_p and β_q are correlated.

- To illustrate, take $p = 10$, $q = 3$, $\beta_p = 10 \cdot \mathbf{1}_p$, $\beta_q = \mathbf{1}_q$, and $\sigma = 10 \cdot \sqrt{R_c}$.

A 'false' model that predicts better than a 'true' one

Setting:

- True (generative) model: $\mathbf{y} = \beta_p \mathbf{X}_p + \beta_q \mathbf{X}_q + \varepsilon$ with $\varepsilon \overset{iid}{\sim} N(\mathbf{0}, \sigma^2 \mathbf{I})$
- Underspecified model: $\mathbf{y} = \beta_p \mathbf{X}_p + \varepsilon$

Theoretical result (Wu, Harris, and McAuley, 2007; Shmueli, 2010):

- An underspecified fit has lower expected MSE when, for $\mathbf{H}_p = \mathbf{X}_p(\mathbf{X}_p^T \mathbf{X}_p)^{-1} \mathbf{X}_p^T$,

$$R_\varepsilon := \beta_q^T \mathbf{X}_q^T (\mathbf{I}_n - \mathbf{H}_p) \mathbf{X}_q \beta_q < q\sigma^2$$

  E.g., when $\|\beta_q\|_1$ is small, σ^2 is large, or β_p and β_q are correlated.

- To illustrate, take $p = 10$, $q = 3$, $\beta_p = 10 \cdot \mathbf{1}_p$, $\beta_q = \mathbf{1}_q$, and $\sigma = 10 \cdot \sqrt{R_\varepsilon}$.

A 'false' model that predicts better than a 'true' one

Setting:

- True (generative) model: $\mathbf{y} = \beta_p \mathbf{X}_p + \beta_q \mathbf{X}_q + \varepsilon$ with $\varepsilon \overset{iid}{\sim} N(\mathbf{0}, \sigma^2 \mathbf{I})$
- Underspecified model: $\mathbf{y} = \beta_p \mathbf{X}_p + \varepsilon$

Theoretical result (Wu, Harris, and McAuley, 2007; Shmueli, 2010):

- An underspecified fit has lower expected MSE when, for $\mathbf{H}_p = \mathbf{X}_p(\mathbf{X}_p^T \mathbf{X}_p)^{-1} \mathbf{X}_p^T$,

$$R_\varepsilon := \beta_q^T \mathbf{X}_q^T (\mathbf{I}_n - \mathbf{H}_p) \mathbf{X}_q \beta_q < q\sigma^2$$

E.g., when $\|\beta_q\|_1$ is small, σ^2 is large, or β_p and β_q are correlated.

- To illustrate, take $p = 10$, $q = 3$, $\beta_p = 10 \cdot \mathbf{1}_p$, $\beta_q = \mathbf{1}_q$, and $\sigma = 10 \cdot \sqrt{R_\varepsilon}$.

A 'false' model that predicts better than a 'true' one

Setting:

- True (generative) model: $\mathbf{y} = \beta_p \mathbf{X}_p + \beta_q \mathbf{X}_q + \varepsilon$ with $\varepsilon \overset{iid}{\sim} N(\mathbf{0}, \sigma^2 \mathbf{I})$
- Underspecified model: $\mathbf{y} = \beta_p \mathbf{X}_p + \varepsilon$

Theoretical result (Wu, Harris, and McAuley, 2007; Shmueli, 2010):

- An underspecified fit has lower expected MSE when, for $\mathbf{H}_p = \mathbf{X}_p(\mathbf{X}_p^T \mathbf{X}_p)^{-1} \mathbf{X}_p^T$,

$$R_c := \beta_q^T \mathbf{X}_q^T (\mathbf{I}_n - \mathbf{H}_p) \mathbf{X}_q \beta_q < q\sigma^2$$

E.g., when $\|\beta_q\|_1$ is small, σ^2 is large, or β_p and β_q are correlated.

- To illustrate, take $p = 10$, $q = 3$, $\beta_p = 10 \cdot \mathbf{1}_p$, $\beta_q = \mathbf{1}_q$, and $\sigma = 10 \cdot \sqrt{R_c}$.

A 'false' model that predicts better than a 'true' one

Setting:

- True (generative) model: $\mathbf{y} = \beta_p \mathbf{X}_p + \beta_q \mathbf{X}_q + \varepsilon$ with $\varepsilon \overset{iid}{\sim} N(\mathbf{0}, \sigma^2 \mathbf{I})$
- Underspecified model: $\mathbf{y} = \beta_p \mathbf{X}_p + \varepsilon$

Theoretical result (Wu, Harris, and McAuley, 2007; Shmueli, 2010):

- An underspecified fit has lower expected MSE when, for $\mathbf{H}_p = \mathbf{X}_p(\mathbf{X}_p^T \mathbf{X}_p)^{-1} \mathbf{X}_p^T$,

$$R_c := \beta_q^T \mathbf{X}_q^T (\mathbf{I}_n - \mathbf{H}_p) \mathbf{X}_q \beta_q < q\sigma^2$$

  E.g., when $\|\beta_q\|_1$ is small, σ^2 is large, or β_p and β_q are correlated.

- To illustrate, take $p = 10$, $q = 3$, $\beta_p = 10 \cdot \mathbf{1}_p$, $\beta_q = \mathbf{1}_q$, and $\sigma = 10 \cdot \sqrt{R_c}$.

A 'false' model that predicts better than a 'true' one

Setting:

- True (generative) model: $\mathbf{y} = \beta_p \mathbf{X}_p + \beta_q \mathbf{X}_q + \varepsilon$ with $\varepsilon \overset{iid}{\sim} N(\mathbf{0}, \sigma^2 \mathbf{I})$
- Underspecified model: $\mathbf{y} = \beta_p \mathbf{X}_p + \varepsilon$

Theoretical result (Wu, Harris, and McAuley, 2007; Shmueli, 2010):

- An underspecified fit has lower expected MSE when, for $\mathbf{H}_p = \mathbf{X}_p(\mathbf{X}_p^T \mathbf{X}_p)^{-1} \mathbf{X}_p^T$,

$$R_c := \beta_q^T \mathbf{X}_q^T (\mathbf{I}_n - \mathbf{H}_p) \mathbf{X}_q \beta_q < q\sigma^2$$

  E.g., when $\|\beta_q\|_1$ is small, σ^2 is large, or β_p and β_q are correlated.

- To illustrate, take $p = 10$, $q = 3$, $\beta_p = 10 \cdot \mathbf{1}_p$, $\beta_q = \mathbf{1}_q$, and $\sigma = 10 \cdot \sqrt{R_c}$.

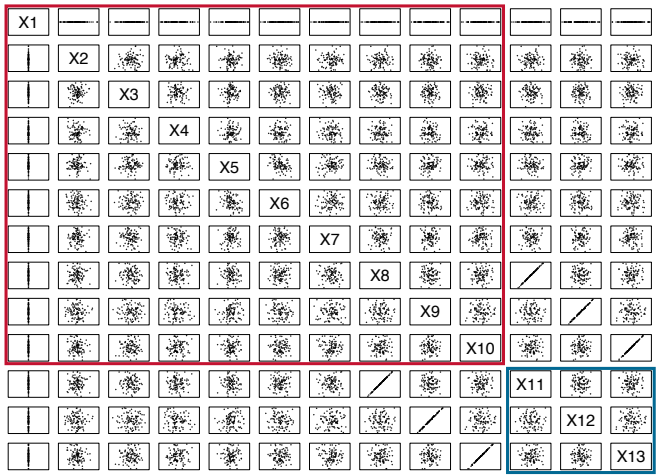
A ‘false’ model that predicts better than a ‘true’ one

A 'false' model that predicts better than a 'true' one

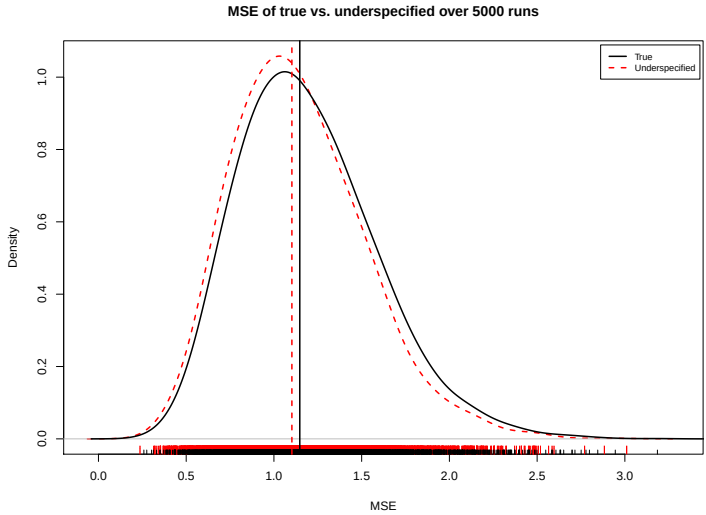
Generate $\mathbf{X} = (\mathbf{X}_p, \mathbf{X}_q)$ so $\mathbf{X}_q$ is correlated with q features in $\mathbf{X}_p$.

A 'false' model that predicts better than a 'true' one

Generate $\mathbf{X} = (\mathbf{X}_p, \mathbf{X}_q)$ so $\mathbf{X}_q$ is correlated with q features in $\mathbf{X}_p$.



A ‘false’ model that predicts better than a ‘true’ one



Selection approach

- In practice, would probably use the lasso, not hand-pick features. What does the lasso give?
- Note: consistency of the lasso is investigated versus the *oracle* predictor (minimizer of L_0 penalty), not the 'truth'
- Conditions involve restrictions on the design matrix (van de Geer and Bühlmann, 2009), which our matrix fails (we think)

Selection approach

- In practice, would probably use the lasso, not hand-pick features. What does the lasso give?
- Note: consistency of the lasso is investigated versus the *oracle* predictor (minimizer of L_0 penalty), not the 'truth'
- Conditions involve restrictions on the design matrix (van de Geer and Bühlmann, 2009), which our matrix fails (we think)

Selection approach

- In practice, would probably use the lasso, not hand-pick features. What does the lasso give?
- Note: consistency of the lasso is investigated versus the *oracle* predictor (minimizer of L_0 penalty), not the 'truth'
- Conditions involve restrictions on the design matrix (van de Geer and Bühlmann, 2009), which our matrix fails (we think)

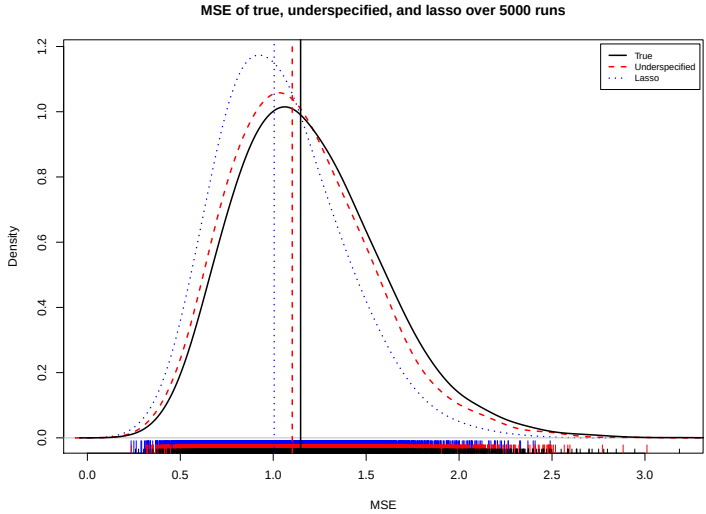
Selection approach

- In practice, would probably use the lasso, not hand-pick features. What does the lasso give?
- Note: consistency of the lasso is investigated versus the *oracle* predictor (minimizer of L_0 penalty), not the 'truth'
- Conditions involve restrictions on the design matrix (van de Geer and Bühlmann, 2009), which our matrix fails (we think)

Selection approach

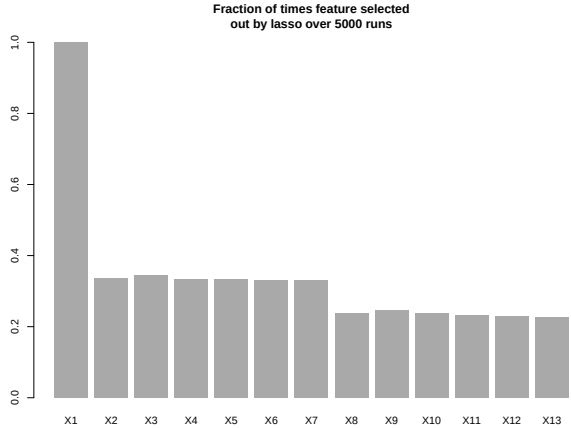
- In practice, would probably use the lasso, not hand-pick features. What does the lasso give?
- Note: consistency of the lasso is investigated versus the *oracle* predictor (minimizer of L_0 penalty), not the 'truth'
- Conditions involve restrictions on the design matrix (van de Geer and Bühlmann, 2009), which our matrix fails (we think)

Selection approach



Selection approach

Interestingly, the λ that is optimal over a validation set always selects out the intercept. Result: neither the true nor the underspecified model!



Bias-variance tradeoff

- The underspecified model and the lasso decrease prediction error versus 'truth' by decreasing the variance
- I.e., the bias-variance tradeoff can explain any disconnect between 'prediction' and 'truth'
- But, bias-variance decomposition is specific to L_2 loss. Does it generalize to arbitrary loss functions?
- No. (Only to strictly convex)

Bias-variance tradeoff

- The underspecified model and the lasso decrease prediction error versus 'truth' by decreasing the variance
- I.e., the bias-variance tradeoff can explain any disconnect between 'prediction' and 'truth'
- But, bias-variance decomposition is specific to L_2 loss. Does it generalize to arbitrary loss functions?
- No. (Only to strictly convex)

Bias-variance tradeoff

- The underspecified model and the lasso decrease prediction error versus 'truth' by decreasing the variance
- I.e., the bias-variance tradeoff can explain any disconnect between 'prediction' and 'truth'
- But, bias-variance decomposition is specific to L_2 loss. Does it generalize to arbitrary loss functions?
- No. (Only to strictly convex)

Bias-variance tradeoff

- The underspecified model and the lasso decrease prediction error versus 'truth' by decreasing the variance
- I.e., the bias-variance tradeoff can explain any disconnect between 'prediction' and 'truth'
- But, bias-variance decomposition is specific to L_2 loss. Does it generalize to arbitrary loss functions?
- No. (Only to strictly convex)

Bias-variance tradeoff

- The underspecified model and the lasso decrease prediction error versus 'truth' by decreasing the variance
- I.e., the bias-variance tradeoff can explain any disconnect between 'prediction' and 'truth'
- But, bias-variance decomposition is specific to L_2 loss. Does it generalize to arbitrary loss functions?
- No. (Only to strictly convex)

Bias-variance tradeoff

- The underspecified model and the lasso decrease prediction error versus 'truth' by decreasing the variance
- I.e., the bias-variance tradeoff can explain any disconnect between 'prediction' and 'truth'
- But, bias-variance decomposition is specific to L_2 loss. Does it generalize to arbitrary loss functions?
- No. (Only to strictly convex)

Generalizing the bias-variance tradeoff

- James (2003) defines a *systematic operator*, S, $S\widehat{f} = \text{argmin}_\mu L(\widehat{f} - \mu)$.
  Mean for L_2, median for L_1
- Generalized variance should depend only on $\widehat{f}$, not on Y
- Generalized variance: $\text{Var}(\widehat{f}) = \mathbb{E}_{\widehat{f}}[L(\widehat{f}, S\widehat{f})]$
- Generalized bias-squared: $\text{bias}^2(Y, \widehat{f}) = L(SY, S\widehat{f})$
- *Effect* of bias and variance are not necessarily equal to these
- *Variance effect* VE compares loss from $\widehat{f}$ to loss from a constant.
  Systematic effect SE compares loss from using $S\widehat{f}$ to loss from using SY

$$\text{VE}(Y, \widehat{f}) = \mathbb{E}_{\widehat{f}, Y}[L(Y, \widehat{f}) - L(Y, S\widehat{f})]$$

$$\text{SE}(Y, \widehat{f}) = \mathbb{E}_{Y}[L(Y, S\widehat{f}) - L(Y, SY)]$$

Generalizing the bias-variance tradeoff

- James (2003) defines a *systematic operator*, S, $S\widehat{f} = \text{argmin}_\mu L(\widehat{f} - \mu)$. Mean for L_2, median for L_1
- Generalized variance should depend only on $\widehat{f}$, not on Y
- Generalized variance: $\text{Var}(\widehat{f}) = \mathbb{E}_{\widehat{f}}[L(\widehat{f}, S\widehat{f})]$
- Generalized bias-squared: $\text{bias}^2(Y, \widehat{f}) = L(SY, S\widehat{f})$
- *Effect* of bias and variance are not necessarily equal to these
- *Variance effect* VE compares loss from $\widehat{f}$ to loss from a constant. *Systematic effect* SE compares loss from using $S\widehat{f}$ to loss from using SY

$$\text{VE}(Y, \widehat{f}) = \mathbb{E}_{\widehat{f}, Y}[L(Y, \widehat{f}) - L(Y, S\widehat{f})]$$

$$\text{SE}(Y, \widehat{f}) = \mathbb{E}_{Y}[L(Y, S\widehat{f}) - L(Y, SY)]$$

Generalizing the bias-variance tradeoff

- James (2003) defines a *systematic operator*, S, $S\widehat{f} = \text{argmin}_\mu L(\widehat{f} - \mu)$. Mean for L_2, median for L_1
- Generalized variance should depend only on $\widehat{f}$, not on Y
- Generalized variance: $\text{Var}(\widehat{f}) = \mathbb{E}_{\widehat{f}}[L(\widehat{f}, S\widehat{f})]$
- Generalized bias-squared: $\text{bias}^2(Y, \widehat{f}) = L(SY, S\widehat{f})$
- *Effect* of bias and variance are not necessarily equal to these
- *Variance effect* VE compares loss from $\widehat{f}$ to loss from a constant. *Systematic effect* SE compares loss from using $S\widehat{f}$ to loss from using SY

$$\text{VE}(Y, \widehat{f}) = \mathbb{E}_{\widehat{f}, Y}[L(Y, \widehat{f}) - L(Y, S\widehat{f})]$$

$$\text{SE}(Y, \widehat{f}) = \mathbb{E}_{Y}[L(Y, S\widehat{f}) - L(Y, SY)]$$

Generalizing the bias-variance tradeoff

- James (2003) defines a *systematic operator*, S, $S\widehat{f} = \text{argmin}_\mu L(\widehat{f} - \mu)$. Mean for L_2, median for L_1
- Generalized variance should depend only on $\widehat{f}$, not on Y
- Generalized variance: $\text{Var}(\widehat{f}) = \mathbb{E}_{\widehat{f}}[L(\widehat{f}, S\widehat{f})]$
- Generalized bias-squared: $\text{bias}^2(Y, \widehat{f}) = L(SY, S\widehat{f})$
- *Effect* of bias and variance are not necessarily equal to these
- *Variance effect* VE compares loss from $\widehat{f}$ to loss from a constant. *Systematic effect* SE compares loss from using $S\widehat{f}$ to loss from using SY

$$\text{VE}(Y, \widehat{f}) = \mathbb{E}_{\widehat{f}, Y}[L(Y, \widehat{f}) - L(Y, S\widehat{f})]$$

$$\text{SE}(Y, \widehat{f}) = \mathbb{E}_{Y}[L(Y, S\widehat{f}) - L(Y, SY)]$$

Generalizing the bias-variance tradeoff

- James (2003) defines a *systematic operator*, S, $S\widehat{f} = \text{argmin}_\mu L(\widehat{f} - \mu)$. Mean for L_2, median for L_1
- Generalized variance should depend only on $\widehat{f}$, not on Y
- Generalized variance: $\text{Var}(\widehat{f}) = \mathbb{E}_{\widehat{f}}[L(\widehat{f}, S\widehat{f})]$
- Generalized bias-squared: $\text{bias}^2(Y, \widehat{f}) = L(SY, S\widehat{f})$
- *Effect* of bias and variance are not necessarily equal to these
- *Variance effect* VE compares loss from $\widehat{f}$ to loss from a constant. *Systematic effect* SE compares loss from using $S\widehat{f}$ to loss from using SY

$$\text{VE}(Y, \widehat{f}) = \mathbb{E}_{\widehat{f},Y}[L(Y, \widehat{f}) - L(Y, S\widehat{f})]$$

$$\text{SE}(Y, \widehat{f}) = \mathbb{E}_{Y}[L(Y, S\widehat{f}) - L(Y, SY)]$$

Generalizing the bias-variance tradeoff

- James (2003) defines a *systematic operator*, S, $S\widehat{f} = \text{argmin}_\mu L(\widehat{f} - \mu)$. Mean for L_2, median for L_1
- Generalized variance should depend only on $\widehat{f}$, not on Y
- Generalized variance: $\text{Var}(\widehat{f}) = \mathbb{E}_{\widehat{f}}[L(\widehat{f}, S\widehat{f})]$
- Generalized bias-squared: $\text{bias}^2(Y, \widehat{f}) = L(SY, S\widehat{f})$
- *Effect* of bias and variance are not necessarily equal to these
- *Variance effect* VE compares loss from $\widehat{f}$ to loss from a constant. *Systematic effect* SE compares loss from using $S\widehat{f}$ to loss from using SY

$$\text{VE}(Y, \widehat{f}) = \mathbb{E}_{\widehat{f},Y}[L(Y, \widehat{f}) - L(Y, S\widehat{f})]$$

$$\text{SE}(Y, \widehat{f}) = \mathbb{E}_{Y}[L(Y, S\widehat{f}) - L(Y, SY)]$$

Generalizing the bias-variance tradeoff

- James (2003) defines a *systematic operator*, S, $S\widehat{f} = \text{argmin}_\mu L(\widehat{f} - \mu)$. Mean for L_2, median for L_1
- Generalized variance should depend only on $\widehat{f}$, not on Y
- Generalized variance: $\text{Var}(\widehat{f}) = \mathbb{E}_{\widehat{f}}[L(\widehat{f}, S\widehat{f})]$
- Generalized bias-squared: $\text{bias}^2(Y, \widehat{f}) = L(SY, S\widehat{f})$
- *Effect* of bias and variance are not necessarily equal to these
- *Variance effect* VE compares loss from $\widehat{f}$ to loss from a constant. *Systematic effect* SE compares loss from using $S\widehat{f}$ to loss from using SY

$$\text{VE}(Y, \widehat{f}) = \mathbb{E}_{\widehat{f},Y}[L(Y, \widehat{f}) - L(Y, S\widehat{f})]$$

$$\text{SE}(Y, \widehat{f}) = \mathbb{E}_{Y}[L(Y, S\widehat{f}) - L(Y, SY)]$$

Generalizing the bias-variance tradeoff

- James (2003) defines a *systematic operator*, S, $S\widehat{f} = \text{argmin}_\mu L(\widehat{f} - \mu)$. Mean for L_2, median for L_1
- Generalized variance should depend only on $\widehat{f}$, not on Y
- Generalized variance: $\text{Var}(\widehat{f}) = \mathbb{E}_{\widehat{f}}[L(\widehat{f}, S\widehat{f})]$
- Generalized bias-squared: $\text{bias}^2(Y, \widehat{f}) = L(SY, S\widehat{f})$
- *Effect* of bias and variance are not necessarily equal to these
- *Variance effect* VE compares loss from $\widehat{f}$ to loss from a constant. *Systematic effect* SE compares loss from using $S\widehat{f}$ to loss from using SY

$$\text{VE}(Y, \widehat{f}) = \mathbb{E}_{\widehat{f},Y}[L(Y, \widehat{f}) - L(Y, S\widehat{f})]$$

$$\text{SE}(Y, \widehat{f}) = \mathbb{E}_{Y}[L(Y, S\widehat{f}) - L(Y, SY)]$$

Applying to 0-1 loss

- Say we have a trained classifier where for some level $X = x$, we have $\mathbb{P}(Y|X = x) = (0.5, 0.4, 0.1)$
- Bayes (optimal) classifier is $f^*(x) = (1, 0, 0)$
- Say that f is not smooth, such that $\widehat{f}$ doesn't manage to be Bayes at x
- Consider two biased classifiers, $\widehat{f}_1(x) = (0, 1, 0)$ and $\widehat{f}_2(x) = (0, 0, 1)$ both with a 'bias' of 1, and zero variance
- What if we randomize the predictions? $\mathbb{P}(\widehat{f}_1(x)|X = x) = (0.4, 0.5, 0.1)$ and $\mathbb{P}(\widehat{f}_2(x)|X = x) = (0.1, 0.5, 0.4)$. Both have the same distribution of probabilities, so, the same variance

Applying to 0-1 loss

- Say we have a trained classifier where for some level $X = x$, we have $\mathbb{P}(Y|X = x) = (0.5, 0.4, 0.1)$
- Bayes (optimal) classifier is $f^*(x) = (1, 0, 0)$
- Say that f is not smooth, such that $\widehat{f}$ doesn't manage to be Bayes at x
- Consider two biased classifiers, $\widehat{f}_1(x) = (0, 1, 0)$ and $\widehat{f}_2(x) = (0, 0, 1)$ both with a 'bias' of 1, and zero variance
- What if we randomize the predictions? $\mathbb{P}(\widehat{f}_1(x)|X = x) = (0.4, 0.5, 0.1)$ and $\mathbb{P}(\widehat{f}_2(x)|X = x) = (0.1, 0.5, 0.4)$. Both have the same distribution of probabilities, so, the same variance

Applying to 0-1 loss

- Say we have a trained classifier where for some level $X = x$, we have $\mathbb{P}(Y|X = x) = (0.5, 0.4, 0.1)$
- Bayes (optimal) classifier is $f^*(x) = (1, 0, 0)$
- Say that f is not smooth, such that $\widehat{f}$ doesn't manage to be Bayes at x
- Consider two biased classifiers, $\widehat{f}_1(x) = (0, 1, 0)$ and $\widehat{f}_2(x) = (0, 0, 1)$ both with a 'bias' of 1, and zero variance
- What if we randomize the predictions? $\mathbb{P}(\widehat{f}_1(x)|X = x) = (0.4, 0.5, 0.1)$ and $\mathbb{P}(\widehat{f}_2(x)|X = x) = (0.1, 0.5, 0.4)$. Both have the same distribution of probabilities, so, the same variance

Applying to 0-1 loss

- Say we have a trained classifier where for some level $X = x$, we have $\mathbb{P}(Y|X = x) = (0.5, 0.4, 0.1)$
- Bayes (optimal) classifier is $f^*(x) = (1, 0, 0)$
- Say that f is not smooth, such that $\widehat{f}$ doesn't manage to be Bayes at x
- Consider two biased classifiers, $\widehat{f}_1(x) = (0, 1, 0)$ and $\widehat{f}_2(x) = (0, 0, 1)$ both with a 'bias' of 1, and zero variance
- What if we randomize the predictions? $\mathbb{P}(\widehat{f}_1(x)|X = x) = (0.4, 0.5, 0.1)$ and $\mathbb{P}(\widehat{f}_2(x)|X = x) = (0.1, 0.5, 0.4)$. Both have the same distribution of probabilities, so, the same variance

Applying to 0-1 loss

- Say we have a trained classifier where for some level $X = x$, we have $\mathbb{P}(Y|X = x) = (0.5, 0.4, 0.1)$
- Bayes (optimal) classifier is $f^*(x) = (1, 0, 0)$
- Say that f is not smooth, such that $\widehat{f}$ doesn't manage to be Bayes at x
- Consider two biased classifiers, $\widehat{f}_1(x) = (0, 1, 0)$ and $\widehat{f}_2(x) = (0, 0, 1)$ both with a 'bias' of 1, and zero variance
- What if we randomize the predictions? $\mathbb{P}(\widehat{f}_1(x)|X = x) = (0.4, 0.5, 0.1)$ and $\mathbb{P}(\widehat{f}_2(x)|X = x) = (0.1, 0.5, 0.4)$. Both have the same distribution of probabilities, so, the same variance

Applying to 0-1 loss

- Say we have a trained classifier where for some level $X = x$, we have $\mathbb{P}(Y|X = x) = (0.5, 0.4, 0.1)$
- Bayes (optimal) classifier is $f^*(x) = (1, 0, 0)$
- Say that f is not smooth, such that $\widehat{f}$ doesn't manage to be Bayes at x
- Consider two biased classifiers, $\widehat{f}_1(x) = (0, 1, 0)$ and $\widehat{f}_2(x) = (0, 0, 1)$ both with a 'bias' of 1, and zero variance
- What if we randomize the predictions? $\mathbb{P}(\widehat{f}_1(x)|X = x) = (0.4, 0.5, 0.1)$ and $\mathbb{P}(\widehat{f}_2(x)|X = x) = (0.1, 0.5, 0.4)$. Both have the same distribution of probabilities, so, the same variance

Applying to 0-1 loss

- Say we have a trained classifier where for some level $X = x$, we have $\mathbb{P}(Y|X = x) = (0.5, 0.4, 0.1)$
- Bayes (optimal) classifier is $f^*(x) = (1, 0, 0)$
- Say that f is not smooth, such that $\widehat{f}$ doesn't manage to be Bayes at x
- Consider two biased classifiers, $\widehat{f}_1(x) = (0, 1, 0)$ and $\widehat{f}_2(x) = (0, 0, 1)$ both with a 'bias' of 1, and zero variance
- What if we randomize the predictions? $\mathbb{P}(\widehat{f}_1(x)|X = x) = (0.4, 0.5, 0.1)$ and $\mathbb{P}(\widehat{f}_2(x)|X = x) = (0.1, 0.5, 0.4)$. Both have the same distribution of probabilities, so, the same variance

Applying to 0-1 loss

- For 0-1 loss,

$$\begin{aligned}\mathrm{VE}(Y,\widehat{f}) &= \mathbb{E}_{\widehat{f},Y}[L(Y,\widehat{f}) - L(Y,S\widehat{f})] = \mathbb{E}_{\widehat{f},Y}[I(Y \neq \widehat{f}) - I(Y \neq S\widehat{f})] \\ &= \mathbb{P}(Y \neq \widehat{f}) - \mathbb{P}(Y \neq S\widehat{f}) = \left(1 - \textstyle\sum_{i=1}^{k} \pi_i \widehat{\pi}_i\right) - \left(1 - \pi_{\mathrm{argmax}_i \widehat{\pi}_i}\right) \\ &= \pi_{\mathrm{argmax}_i \widehat{\pi}_i} - \textstyle\sum_{i=1}^{k} \pi_i \widehat{\pi}_i\end{aligned}$$

- Then,

$$\mathrm{VE}(\widehat{f}_1, Y) = 0.4 - (0.5 \cdot 0.4 + 0.4 \cdot 0.5 + 0.1 \cdot 0.1) = .40 - 0.41 = \boxed{-0.01}$$

$$\mathrm{VE}(\widehat{f}_2, Y) = 0.4 - (0.5 \cdot 0.1 + 0.4 \cdot 0.5 + 0.1 \cdot 0.4) = .40 - 0.29 = \boxed{0.11}$$

- The same *amount* of variance can increase or decrease the expected accuracy, completely separate from the bias!

Applying to 0-1 loss

- For 0-1 loss,

$$\begin{aligned}\mathrm{VE}(Y,\widehat{f}) &= \mathbb{E}_{\widehat{f},Y}[L(Y,\widehat{f}) - L(Y,S\widehat{f})] = \mathbb{E}_{\widehat{f},Y}[I(Y \neq \widehat{f}) - I(Y \neq S\widehat{f})] \\ &= \mathbb{P}(Y \neq \widehat{f}) - \mathbb{P}(Y \neq S\widehat{f}) = \left(1 - \textstyle\sum_{i=1}^{k} \pi_i \widehat{\pi}_i\right) - \left(1 - \pi_{\mathrm{argmax}_i \widehat{\pi}_i}\right) \\ &= \pi_{\mathrm{argmax}_i \widehat{\pi}_i} - \textstyle\sum_{i=1}^{k} \pi_i \widehat{\pi}_i\end{aligned}$$

- Then,

$$\mathrm{VE}(\widehat{f}_1, Y) = 0.4 - (0.5 \cdot 0.4 + 0.4 \cdot 0.5 + 0.1 \cdot 0.1) = .40 - 0.41 = \boxed{-0.01}$$

$$\mathrm{VE}(\widehat{f}_2, Y) = 0.4 - (0.5 \cdot 0.1 + 0.4 \cdot 0.5 + 0.1 \cdot 0.4) = .40 - 0.29 = \boxed{0.11}$$

- The same *amount* of variance can increase or decrease the expected accuracy, completely separate from the bias!

Applying to 0-1 loss

- For 0-1 loss,

$$\begin{aligned}\mathrm{VE}(Y,\widehat{f}) &= \mathbb{E}_{\widehat{f},Y}[L(Y,\widehat{f}) - L(Y,S\widehat{f})] = \mathbb{E}_{\widehat{f},Y}[I(Y \neq \widehat{f}) - I(Y \neq S\widehat{f})] \\ &= \mathbb{P}(Y \neq \widehat{f}) - \mathbb{P}(Y \neq S\widehat{f}) = \left(1 - \textstyle\sum_{i=1}^{k} \pi_i \widehat{\pi}_i\right) - \left(1 - \pi_{\mathrm{argmax}_i \widehat{\pi}_i}\right) \\ &= \pi_{\mathrm{argmax}_i \widehat{\pi}_i} - \textstyle\sum_{i=1}^{k} \pi_i \widehat{\pi}_i\end{aligned}$$

- Then,

$$\mathrm{VE}(\widehat{f}_1, Y) = 0.4 - (0.5 \cdot 0.4 + 0.4 \cdot 0.5 + 0.1 \cdot 0.1) = .40 - 0.41 = \boxed{-0.01}$$

$$\mathrm{VE}(\widehat{f}_2, Y) = 0.4 - (0.5 \cdot 0.1 + 0.4 \cdot 0.5 + 0.1 \cdot 0.4) = .40 - 0.29 = \boxed{0.11}$$

- The same *amount* of variance can increase or decrease the expected accuracy, completely separate from the bias!

Applying to 0-1 loss

- For 0-1 loss,

$$\begin{aligned}\mathrm{VE}(Y,\widehat{f}) &= \mathbb{E}_{\widehat{f},Y}[L(Y,\widehat{f}) - L(Y,S\widehat{f})] = \mathbb{E}_{\widehat{f},Y}[I(Y \neq \widehat{f}) - I(Y \neq S\widehat{f})] \\ &= \mathbb{P}(Y \neq \widehat{f}) - \mathbb{P}(Y \neq S\widehat{f}) = \left(1 - \textstyle\sum_{i=1}^{k} \pi_i \widehat{\pi}_i\right) - \left(1 - \pi_{\mathrm{argmax}_i \widehat{\pi}_i}\right) \\ &= \pi_{\mathrm{argmax}_i \widehat{\pi}_i} - \textstyle\sum_{i=1}^{k} \pi_i \widehat{\pi}_i\end{aligned}$$

- Then,

$$\mathrm{VE}(\widehat{f}_1, Y) = 0.4 - (0.5 \cdot 0.4 + 0.4 \cdot 0.5 + 0.1 \cdot 0.1) = .40 - 0.41 = \boxed{-0.01}$$

$$\mathrm{VE}(\widehat{f}_2, Y) = 0.4 - (0.5 \cdot 0.1 + 0.4 \cdot 0.5 + 0.1 \cdot 0.4) = .40 - 0.29 = \boxed{0.11}$$

- The same *amount* of variance can increase or decrease the expected accuracy, completely separate from the bias!

Applying to 0-1 loss

- For 0-1 loss,

$$\begin{aligned}\mathrm{VE}(Y,\widehat{f}) &= \mathbb{E}_{\widehat{f},Y}[L(Y,\widehat{f}) - L(Y,S\widehat{f})] = \mathbb{E}_{\widehat{f},Y}[I(Y \neq \widehat{f}) - I(Y \neq S\widehat{f})] \\ &= \mathbb{P}(Y \neq \widehat{f}) - \mathbb{P}(Y \neq S\widehat{f}) = \left(1 - \textstyle\sum_{i=1}^{k} \pi_i \widehat{\pi}_i\right) - \left(1 - \pi_{\mathrm{argmax}_i \widehat{\pi}_i}\right) \\ &= \pi_{\mathrm{argmax}_i \widehat{\pi}_i} - \textstyle\sum_{i=1}^{k} \pi_i \widehat{\pi}_i\end{aligned}$$

- Then,

$$\mathrm{VE}(\widehat{f}_1, Y) = 0.4 - (0.5 \cdot 0.4 + 0.4 \cdot 0.5 + 0.1 \cdot 0.1) = .40 - 0.41 = \boxed{-0.01}$$

$$\mathrm{VE}(\widehat{f}_2, Y) = 0.4 - (0.5 \cdot 0.1 + 0.4 \cdot 0.5 + 0.1 \cdot 0.4) = .40 - 0.29 = \boxed{0.11}$$

- The same *amount* of variance can increase or decrease the expected accuracy, completely separate from the bias!

Take-aways

- Strangely enough, models that predict the best are not necessarily 'true' (even in terms of associations/correlations)
- The bias-variance tradeoff can help us understand how this is possible for L_2 loss, but other loss functions can be even more strange
- Prediction and explanation are very different tasks!
- Understanding and communicating how exactly this plays out is important for any application of statistics and machine learning

Take-aways

- Strangely enough, models that predict the best are not necessarily 'true' (even in terms of associations/correlations)
- The bias-variance tradeoff can help us understand how this is possible for L_2 loss, but other loss functions can be even more strange
- Prediction and explanation are very different tasks!
- Understanding and communicating how exactly this plays out is important for any application of statistics and machine learning

Take-aways

- Strangely enough, models that predict the best are not necessarily 'true' (even in terms of associations/correlations)
- The bias-variance tradeoff can help us understand how this is possible for L_2 loss, but other loss functions can be even more strange
- Prediction and explanation are very different tasks!
- Understanding and communicating how exactly this plays out is important for any application of statistics and machine learning

Take-aways

- Strangely enough, models that predict the best are not necessarily 'true' (even in terms of associations/correlations)
- The bias-variance tradeoff can help us understand how this is possible for L_2 loss, but other loss functions can be even more strange
- Prediction and explanation are very different tasks!
- Understanding and communicating how exactly this plays out is important for any application of statistics and machine learning

Take-aways

- Strangely enough, models that predict the best are not necessarily 'true' (even in terms of associations/correlations)
- The bias-variance tradeoff can help us understand how this is possible for L_2 loss, but other loss functions can be even more strange
- Prediction and explanation are very different tasks!
- Understanding and communicating how exactly this plays out is important for any application of statistics and machine learning

References

Box, George E. P. (1979). *Robustness in the strategy of scientific model building*. Tech. rep. #1954. University of Madison-Wisconsin, Mathematics Research Center.

Breiman, Leo (2001). "Statistical modeling: The two cultures (with comments and a rejoinder by the author)". In: *Statistical Science* 16.3, pp. 199–231. DOI: 10.1214/ss/1009213726.

James, Gareth M. (2003). "Variance and bias for general loss functions". In: *Machine Learning* 51.2, pp. 115–135. DOI: 10.1023/A:1022899518027.

Shmueli, Galit (2010). "To explain or to predict?" In: *Statistical Science* 25.3, pp. 289–310. DOI: 10.1214/10-STS330.

van de Geer, Sara A. and Peter Bühlmann (2009). "On the conditions used to prove oracle results for the lasso". In: *Electronic Journal of Statistics* 3, pp. 1360–1392. DOI: 10.1214/09-EJS506.

Wu, Shaohua, T. J. Harris, and K. B. McAuley (2007). "The use of simplified or misspecified models: Linear case". In: *The Canadian Journal of Chemical Engineering* 85.4, pp. 386–398. DOI: 10.1002/cjce.5450850401.

Statistical Topological Data Analysis

Chaitanya Ahuja[1]
Bhuwan Dhingra[1]

[1]Language Technologies Institute
Carnegie Mellon University

Contents

Topological Data Analysis

- **Topological Data Analysis (TDA)** refers to data analysis methods which study properties such as shape, topology and connectedness of the data.
- This includes:
  - Clustering (particularly Density Based Clustering)
  - Density Modes and Ridge Estimation
  - Manifold Learning / Dimension Reduction
  - Persistent Homology
- TDA is useful as a visualization tool and for summarizing high-dimensional datasets.

This Project

- We review recent work [1] on performing statistical inference for *Density Trees*—a particular class of hierarchical clustering methods.
- Outline:
  - Definitions and Tree Topology
  - Constructing confidence sets via bootstrap
  - Pruning trees to remove insignificant features
- As an application, we generate density trees to visualize distribution of words in documents

Density Trees

Suppose the data lies in $\mathcal{X} \subset \mathbb{R}^d$. Given a density function $f : \mathcal{X} \to [0, \infty)$,

- Let $T_f(\lambda)$ denote the connected components of the upper level set $\{x : f(x) > \lambda\}$. These are the high density clusters at level λ.
- The density tree is the collection of all such clusters: $\{T_f\} = T_f = \cup_\lambda T_f(\lambda)$.
- This is a tree by construction, i.e. if $A, B \in \{T_f\}$, then either $A \subset B$, or $B \subset A$ or $A \cap B = \phi$.

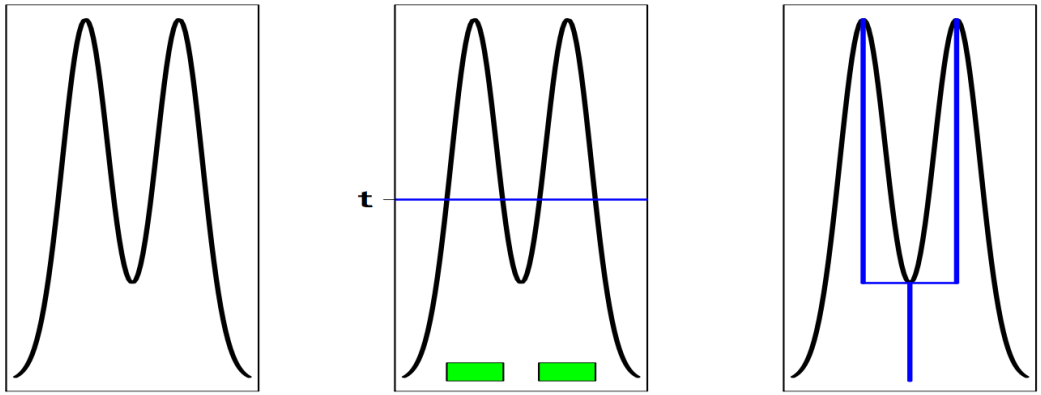


Figure: Obtained from [2]

Estimated Tree

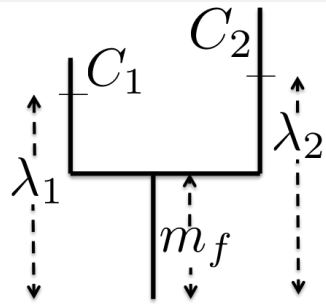
In general we have an iid sample from the true density $X_1, X_2, \ldots, X_N \sim p$. The **Estimated Tree** $T_{\hat{p}_h}$ is the tree constructed from the Kernel Density Estimate:

$$\hat{p}_h(x) = \frac{1}{nh^d} \sum_{i=1}^{N} K(\frac{\|x - X_i\|}{h})$$

$$T_{\hat{p}_h}(\lambda) = \{x : \hat{p}_h(x) > \lambda\}$$

Tree Topology

- Given a tree $\{T_f\}$, we can define the **tree distance function** between elements of the tree:



$$d_{T_f}(C_1, C_2) = \lambda_1 + \lambda_2 - 2m_f(C_1, C_2) \qquad C_1, C_2 \in \{T_f\}$$

- It can be shown that d_{T_f} is a metric on $\{T_f\}$, and hence induces a metric topology on it.

Lemma

If the true unknown density p is a morse function, then $\exists$ a constant $h_0 > 0$, such that $\forall h$ s.t. $0 < h \leq h_0$, the true cluster tree, T_p and the estimated tree $T_{\hat{p}_h}$ have the same metric topology above.

Hence we do not need to let the KDE bandwidth $h \to 0$. This leads to a dimension-independent rate of convergence for the bootstrap confidence set.

Confidence Sets via Bootstrap

- To construct confidence sets, we first need a metric to measure the "closeness" of two trees. The l_∞ metric is defined as,

$$d_\infty(T_p, T_q) = \sup_{x \in \chi} |p(x) - q(x)| = \|p - q\|_\infty$$

- The confidence set is defined as $C_\alpha = \{T : d_\infty(T, T_{\hat{p}_h}) \leq t_\alpha\}$ for T_{p_h}.
- t_α can be obtained by the bootstrap:

$$\begin{aligned} \hat{F}(s) &= \frac{1}{B} \sum_{i=1}^{B} \mathbb{I}(d_\infty(\tilde{T}^i_{p_h}, T_{\hat{p}_h}) < s) \\ \hat{t_\alpha} &= \hat{F}^{-1}(1-\alpha) \end{aligned}$$

Where $\{\tilde{T}^1_{p_h}, \ldots, \tilde{T}^B_{p_h}\}$ are the estimated trees for the bootstrap samples $\{\tilde{X}^1_1, \ldots, \tilde{X}^1_n\}, \ldots, \{\tilde{X}^B_1, \ldots, \tilde{X}^B_n\}$.

Convergence Rate

Theorem

Under regularity conditions on the kernel, the constructed confidence interval is asymptotically valid and satisfies,

$$\mathbb{P}\left(T_p \in \hat{C}_\alpha\right) = 1 - \alpha + O\left(\frac{\log^7 n}{nh^d}\right)^{\frac{1}{6}} \tag{1}$$

where $\hat{C}_\alpha = \{T : d_\infty(T, T_{\hat{p}_h}) \leq \hat{t}_\alpha\}$

From the Lemma presented previously, we can fix h to a small constant, to obtain a dimension-independent rate of $O\left(\frac{\log^7 n}{n}\right)^{\frac{1}{6}}$.

Notions of Tree Simplicity

- The confidence set $\hat{C}_\alpha$, contains infinitely many trees—including very complex ones obtained by small perturbations of the density estimate.
- We would like to obtain "simple" trees by removing statistically insignificant features.
- A notion of simplicity is given by the following partial ordering:

Definition

For any $f, g : \mathcal{X} \to [0, \infty)$ and their trees T_f, T_g we say $T_f \preceq T_g$ if $\exists$ a map $\Phi : \{T_f\} \to \{T_g\}$ which preserves set inclusion relationships, i.e. for any $C_1, C_2 \in \{T_f\}$ we have $C_1 \subset C_2$ iff $\Phi(C_1) \subset \Phi(C_2)$.

- This partial ordering matches intuitive notions of simplicity, for e.g. if T_f is obtained by removing edges from T_g, then $T_f \preceq T_g$.

Pruning Rules

Following two strategies are suggested to prune the empirical tree $T_{\hat{p}_h}$:

1. **Pruning leaves:** Remove all leaves of the tree with length less than $2\hat{t}_\alpha$.
2. **Pruning leaves and internal branches:** Remove all leaves and internal branches of the tree with *cumulative length* less than $2\hat{t}_\alpha$.

It can be shown that the tree obtained after pruning from either of these two strategies,

- Is simpler than $T_{\hat{p}_h}$.
- Is generated from a valid density function.
- And the density function lies in the constructed confidence set.

Visualization of Word Embeddings

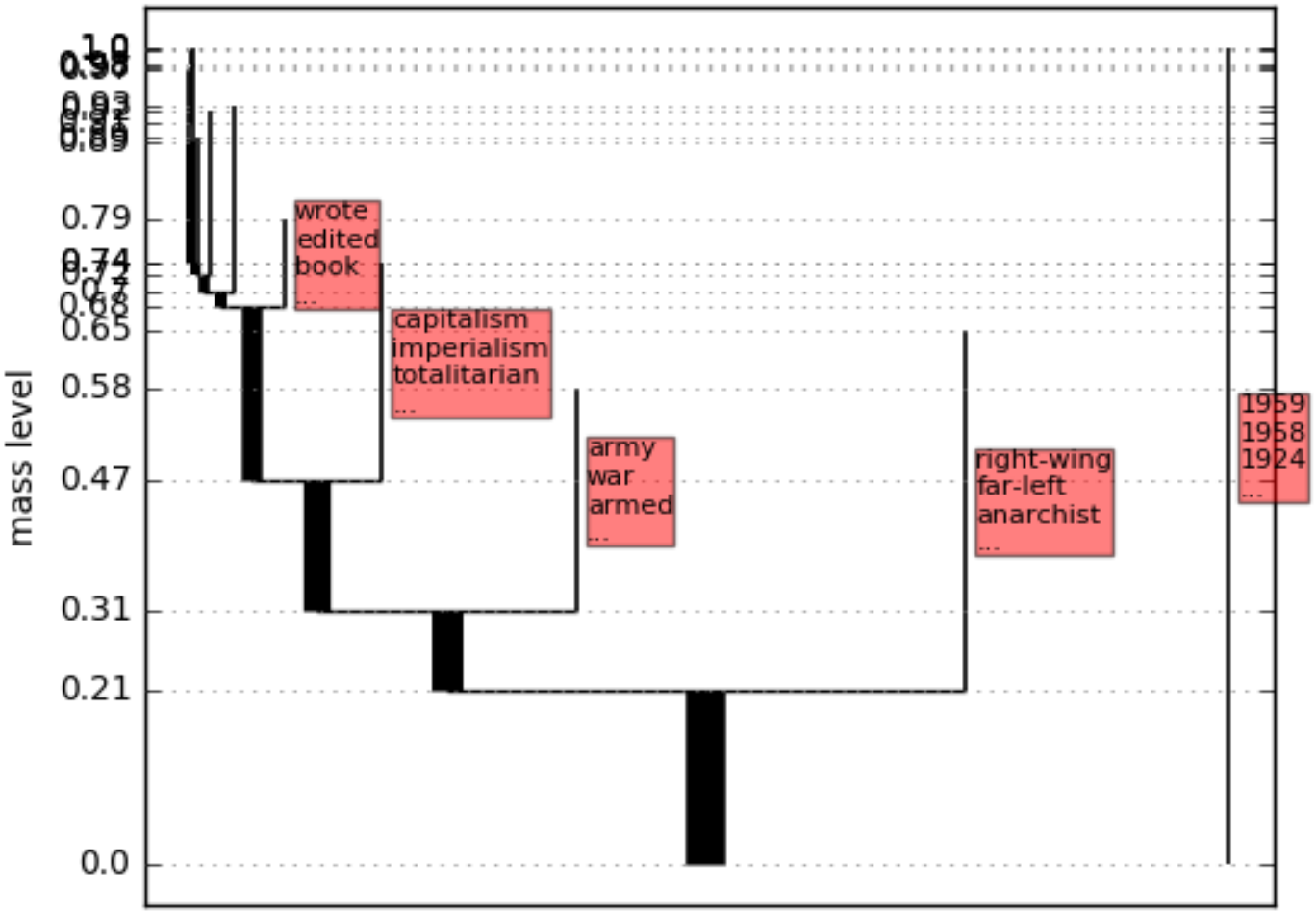


Figure: Cluster tree for Wikipedia Page on **Noam Chomsky**

Visualization of Word Embeddings

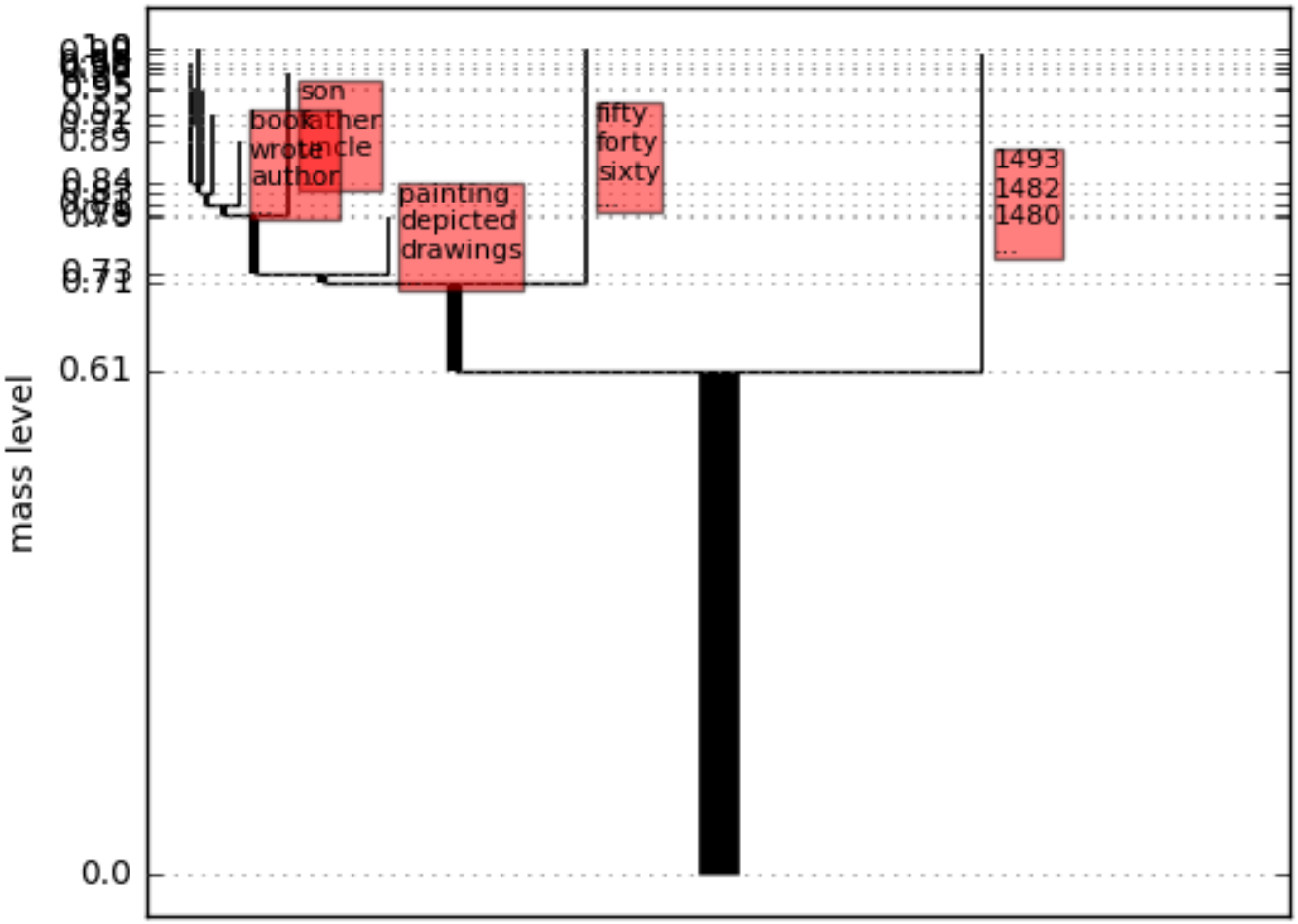


Figure: Cluster tree for Wikipedia Page on **Leonardo da Vinci**

References I


Jisu Kim et al. "Statistical Inference for Cluster Trees". In: *arXiv preprint arXiv:1605.06416* (2016).


Larry Wasserman. "Topological Data Analysis". In: *arXiv preprint arXiv:1609.08227* (2016).

PCA-Based estimation for functional linear regression with functional responses

Ruixi Fan Shuo Zhao

Carnegie Mellon University, Pittsburgh, PA

CCML, 2017

Outline

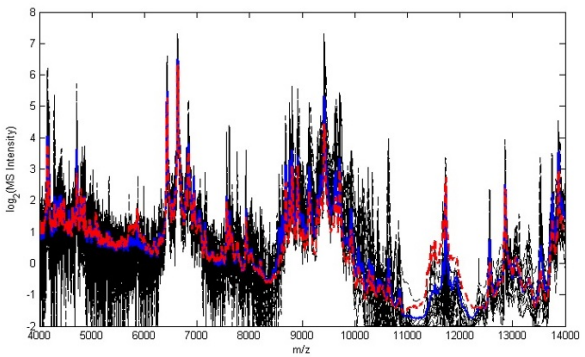
1. Introduction
   - Functional data analysis (FDA)
   - Functional linear regression & Motivation
   - Functional Principal Component Analysis(FPCA)
   - Problem statement

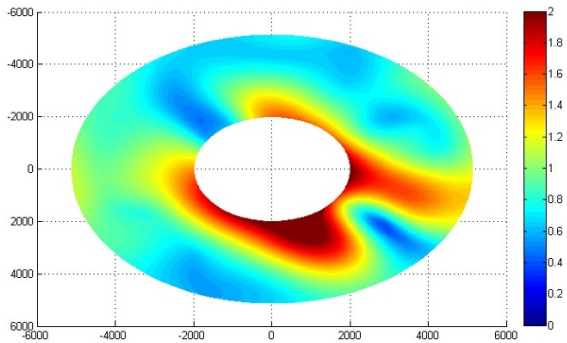
2. Proof of convergence rates
   - Divide the problem
   - Bound the error terms
   - Minimax rate

Outline

Functional data analysis (FDA)

- Functional Data (FD) refer to data recorded during a time interval or intermittently at several discrete time points.
- Functional Data Analysis (FDA) deals with FD for classification, clustering, regression etc. In FDA, each sample element is considered to be a function over time, spatial location, wavelength, probability and so on.
- Applications: time series, images, shapes, or more general objects.





Outline

Motivation

- Functional predictor regression
- Functional response regression
- Function-on-function regression

- Estimation of the coefficient function
- Optimal convergence rate

Outline

Functional Principal Component Analysis(FPCA)

Covariance function

$$K(s,t) = Cov\{X(s), X(t)\}$$

Spectral expansion of covariance

According to the Hilbert-Schmidt theorem,

$$K(s,t) = \sum_{k=1}^{\infty} \kappa_k \phi_k(s) \phi_k(t)$$

where $\kappa_1 \geq \kappa_2 \geq ... > 0$. It has

Functional Principal Component Analysis(FPCA)

Spectral expansion and FPCA

$$X(t) = \mu_X(t) + \sum_{k=1}^{\infty} \xi_k \phi_k(t)$$

where all the coefficients are listed in the order: $\xi_1 > \xi_2 > ... > 0$, then truncate at $k = m$,

$$X(t) \approx X_m(t) = \mu_X(t) + \sum_{k=1}^{m} \xi_k \phi_k(t)$$

Outline

Functional linear regression

Functional linear regression

$$Y(s) = \mu_Y(s) + \int_I b(s,t)[X(t) - \mu_X(t)]dt + \epsilon(t)$$

where $I = [0, 1]$.

Regression model by expectation

$$E(Y|X)(s) = \mu_Y(s) + \int_I b(s,t)[X(t) - \mu_X(t)]dt$$

where $E(Y|X)$ is the conditional expectation of Y as an $L^2(I)$-valued random variable conditionally on the σ-field generated by X.

Formulation of estimator

Single Truncation estimator

$$\hat{b}(s,t) = \sum_{k=1}^{m_n} \frac{1}{\hat{\kappa}_k} (\frac{1}{n} \sum \hat{\xi}_{i,k} Y_i(s)) \hat{\phi}_k(t)$$

where $K(s,t)$ is estimated in the emprical covariance function:
$\hat{K(s,t)} = \frac{1}{n} \sum_{i=1}^{n} (X_i(s) - \bar{X}(s))(X_i(t) - \bar{X}(t))$.
Expand $\hat{K}(s,t)$ with eigenvalue $\{\hat{\kappa}_k\}_{k=1}^{\infty}$ and eigenfunctions $\{\hat{\phi}_k\}_{k=1}^{\infty}$.

$$\hat{\xi}_{i,k} = \inf_{I} \{X_i(t) - \bar{X}(t)\} \phi_k(t) dt$$

Convergence rates

General Assumptions

$\exists \quad \alpha > 1, \beta > 1.5, \gamma > 0.5, C_1 > 0$ st

$$E[||X||^2] < \infty, E[||Y||^2|X] \leq C_1 a.s.; \quad E[\xi_k^4] \leq C_1 \kappa_k^2, \forall k \geq 1;$$

$$\kappa_k \leq C_1 k^{-\alpha}, \kappa_k - \kappa_{k+1} \geq C_1^{-1} k^{-\alpha-1}, \forall k \geq 1;$$

$$|b_{j,k}| \leq C_1 j^{-\gamma} k^{-\beta}, \forall j, k \geq 1, \beta > \frac{\alpha}{2} + 1;$$

Convergence rate of Single truncation

$\exists \quad \alpha > 1, \beta > 1.5, \gamma > 0.5, C_1 > 0$, choose m_n such that $m_n = o(n^{1/(2\alpha+2)})$, then

$$|||\hat{b} - b|||^2 = O_p(n^{-(2\beta-1)/(\alpha+2\beta)})$$

Outline

Some important expansions

On $\{\phi_j \otimes \hat{\phi}_k\}_{j,k=1}^{\infty}$, expand b and $\hat{b}$:

$$b = \sum_{j,k} \dot{b}_{j,k}(\phi_j \otimes \hat{\phi}_k)$$

$$\hat{b} = \sum_{j=1}^{\infty}\sum_{k=1}^{m_n} \hat{b}_{j,k}(\phi_j \otimes \hat{\phi}_k)$$

Note: $\phi_j \otimes \hat{\phi}_k \equiv \phi_j(s)\hat{\phi}_k(t)$, $\hat{b}_{j,k} = \frac{n^{-1}\sum_{i=1}^{n}\eta_{i,j}\hat{\xi}_{i,k}}{\hat{\kappa}_k}$,
$\dot{b}_{j,k} = \iint b(s,t)(\phi_j \otimes \hat{\phi}_k)$, $b_{j,k} = \int\int_{I^2} b(s,t)(\phi_j \otimes \phi_k)$
Set $\eta^c_{i,j} = \eta_{i,j} - \frac{\sum_{k=1}^{n}\eta_{k,j}}{n}$ and $\epsilon^c_{i,j} = \epsilon_{i,j} - \frac{\sum_{k=1}^{n}\epsilon_{k,j}}{n}$,
$\Rightarrow \eta^c_{i,j} = \sum_l \dot{b}_{j,l}\hat{\xi}_{i,l} + \epsilon^c_{i,j}$.

$$\Rightarrow \quad \hat{b}_{j,k} = \frac{1}{\hat{\kappa}_k}\left(\frac{1}{n}\sum_{i=1}^{n}\sum_{l}\dot{b}_{j,l}\hat{\xi}_{i,l}\hat{\xi}_{i,k} + \frac{1}{n}\sum_{i=1}^{n}\epsilon_{i,j}\hat{\xi}_{i,k}\right) = \dot{b}_{j,k} + \frac{1}{\hat{\kappa}_k}\left(\frac{1}{n}\sum_{i=1}^{n}\epsilon_{i,j}\hat{\xi}_{i,k}\right)$$

.

Rewrite the difference $\hat{b} - b$

Break down $\hat{b} - b$:

$$\hat{b} - b = \sum_{j=1}^{\infty}\sum_{k=1}^{m_n}(\hat{b}_{j,k} - b_{j,k})(\phi_j \otimes \hat{\phi}_k) + \sum_{j=1}^{\infty}\sum_{k=1}^{m_n} b_{j,k}\{\phi_j \otimes (\hat{\phi}_k - \phi_k)\}$$
$$+ (\sum_{j=1}^{\infty}\sum_{k>m_n} b_{j,k}(\phi_j \otimes \phi_k))$$

Rewrite $|||\hat{b} - b|||^2$

$$|||\hat{b} - b|||^2 \lesssim \sum_{j=1}^{\infty}\sum_{k=1}^{m_n}(\hat{b}_{j,k} - b_{j,k})^2 + \iint\{\sum_{j=1}^{\infty}\sum_{k=1}^{m_n} b_{j,k}\{\phi_j \otimes (\hat{\phi}_k - \phi_k)\}\}^2$$

$$+ \sum_{j=1}^{\infty}\sum_{k>m_n} b_{j,k}^2$$

$$\lesssim \underbrace{\sum_{j=1}^{\infty}\sum_{k=1}^{m_n}\frac{1}{\hat{\kappa}_k^2}(\frac{1}{n}\sum_{i=1}^{n}\epsilon_{i,j}\hat{\xi}_{i,k})^2}_{\text{eigenvalue error}} + \underbrace{\sum_{j=1}^{\infty}\sum_{k=1}^{m_n}(\dot{b}_{j,k} - b_{j,k})^2}_{\text{coefficient error}}$$

$$+ \underbrace{\iint\{\sum_{j=1}^{\infty}\sum_{k=1}^{m_n} b_{j,k}\{\phi_j \otimes (\hat{\phi}_k - \phi_k)\}\}^2}_{\text{basis error}} + \underbrace{\sum_{j=1}^{\infty}\sum_{k>m_n} b_{j,k}^2}_{\text{higher-order term}}$$

Outline

Bound the eigenvalue error

To bound $\sum_{j=1}^{\infty}\sum_{k=1}^{m_n}\frac{1}{\hat{\kappa}_k^2}(\frac{1}{n}\sum_{i=1}^{n}\epsilon_{i,j}\hat{\xi}_{i,k})^2$, we have

$$|\hat{\kappa}_k/\kappa_k - 1| \lesssim k^{\alpha}|\hat{\kappa}_k - \kappa_k| \lesssim m_n^{\alpha}|||\hat{K} - K||| = O_p(1)$$

Since $\epsilon_{i,j}$ are independent with mean zero conditionally on $X_1^n = \{X_1, ..., X_n\}$, we get

$$E[(\frac{1}{n}\sum_{i=1}^{n}\epsilon_{i,j}\hat{\xi}_{i,k})^2|X_1^n] = \frac{1}{n^2}\sum_{i=1}^{n}E[\epsilon_{i,j}^2|X_1^n]\hat{\xi}_{i,k}^2$$

With Bessel's inequality :
$\sum_{j=1}^{\infty}E[\epsilon_{i,j}^2|X_1^n] = E[\sum_{j=1}^{\infty}\epsilon_{i,j}^2|X_1^n] \leq E[||\mathcal{E}_i||^2|X_1^n] \leq C_1$, we have

$$E[\sum_{j=1}^{\infty}\sum_{k=1}^{m_n}\frac{1}{\hat{\kappa}_k^2}(\frac{1}{n}\sum_{i=1}^{n}\epsilon_{i,j}\hat{\xi}_{i,k})^2|X_1^n] \lesssim \frac{1}{n}\sum_{k=1}^{m_n}\frac{1}{\hat{\kappa}_k} = O_p(n^{-1}m_n^{\alpha+1})$$

$$\Rightarrow \sum_{j=1}^{\infty}\sum_{k=1}^{m_n}\frac{1}{\hat{\kappa}_k^2}(\frac{1}{n}\sum_{i=1}^{n}\epsilon_{i,j}\hat{\xi}_{i,k})^2 = O_p(n^{-1}m_n^{\alpha+1})$$

Decompose the coefficient error

To bound $\sum_{j=1}^{\infty}\sum_{k=1}^{m_n}(\dot{b}_{j,k}-b_{j,k})^2$, generally we assume that $\inf_{l:l\neq k}|\hat{\kappa}_k-\kappa_l|>0$, then

$$\hat{\phi}_k-\phi_k=\sum_{l:l\neq k}(\hat{\kappa}_k-\kappa_l)^{-1}\phi_l\int\int(\hat{K}-K)(\hat{\phi}_k\otimes\phi_l)+\phi_k\int(\hat{\phi}_k-\phi_k)\phi_k$$

$$\begin{aligned}
\dot{b}_{j,k}-b_{j,k}&=\sum_{l:l\neq k}b_{j,l}(\hat{\kappa}_k-\kappa_l)^{-1}\int\int(\hat{K}-K)(\phi_k\otimes\phi_l)\\
&\quad+\sum_{l:l\neq k}b_{j,l}\{(\hat{\kappa}_k-\kappa_l)^{-1}-(\kappa_k-\kappa_l)^{-1}\}\int\int(\hat{K}-K)(\phi_k\otimes\phi_l)\\
&\quad+\sum_{l:l\neq k}b_{j,l}(\hat{\kappa}_k-\kappa_l)^{-1}\int\int(\hat{K}-K)((\hat{\phi}_k-\phi_k)\otimes\phi_l)\\
&\quad+b_{j,k}\int(\hat{\phi}_k-\phi_k)\phi_k\\
&:=T_{j,k,1}+T_{j,k,2}+T_{j,k,3}+T_{j,k,4}
\end{aligned}$$

bound component $T_{j,k,4}$, $T_{j,k,3}$

It is clear that $|T_{j,k,4}| \lesssim j^{-\gamma} k^{-\beta} ||\hat{\phi}_k - \phi_k||$.
Since $|\int\int(\hat{K} - K)\{(\hat{\phi}_k - \phi_k) \otimes \phi_l\}| \leq |||\hat{K} - K|||\dot{|}|\hat{\phi}_k - \phi_k||$,
we will get the following:
$|T_{j,k,3}| \lesssim j^{-\gamma} |||\hat{K} - K||| \cdot ||\hat{\phi}_k - \phi_k|| \sum_{l:l\neq k} \frac{l^{-\beta}}{|\kappa_k - \kappa_l|}$

bound $T_{j,k,1}$

Based on the assumption that $\kappa_k \simeq k^{-\alpha}$, we can choose $k \geq 1$ and $C > 1$ large enough so that $\kappa_k/\kappa_{k/C} \leq 1/2$ and $\kappa_{[Ck]+1}/\kappa_k \leq 1/2, \forall k \geq k_0$, where [.] is a ceiling function. Now, we can partition the sum into three parts: $\sum_{l:l\neq k} = \sum_{l=1}^{[k/C]} + \sum_{l=[k/C]+1}^{[Ck]} + \sum_{l=[Ck]+1}^{\infty}$
Through suitable estimation, we can get the following:

$$\sum_{l=1}^{[k/C]} \frac{l^{-\beta}}{(\kappa_l - \kappa_k)} \lesssim \begin{cases} 1, & \text{if } \beta > \alpha + 1 \\ \log k, & \text{if } \beta = \alpha + 1 \\ k^{\alpha-\beta+1}, & \text{if } \beta < \alpha + 1 \end{cases}$$

$$\sum_{l=[Ck]+1}^{\infty} \frac{l^{-\beta}}{(\kappa_k - \kappa_l)} \lesssim k^{\alpha-\beta+1}$$

$$\sum_{[k/C]+1}^{[Ck]} \frac{l^{-\beta}}{|\kappa_k - \kappa_l|} \lesssim \begin{cases} 1, & \beta > \alpha + 1 \\ k^{\alpha-\beta+1} \log k, & \beta \leq \alpha + 1 \end{cases}$$

$$\Rightarrow E(T_{j,k,1}^2) \lesssim n^{-1} j^{-2\gamma} k^{-\alpha}$$

bound $T_{j,k,2}$

Define the event A_n:

$$A_n = \{|\hat{\kappa}_k - \kappa_l| \geq |\kappa_k - \kappa_l|/2, \forall k : 1 \leq k \leq m_n, \forall l \neq k\}$$

On the event A_n,

$$|T_{j,k,2}| \lesssim j^{-\gamma} |||\hat{K} - K||| \sum_{l:l\neq k} \frac{l^{-\beta}\hat{v}_{k,l}}{|\kappa_k - \kappa_l|^2}$$

where $\hat{v}_{k,l} = |\frac{1}{n}\sum_{i=1}^n \xi_{i,k}\xi_{i,l} - \bar{\xi}_k\bar{\xi}_l|$, then we have

$$E\{(\sum_{l:l\neq k} \frac{l^{-\beta}}{|\kappa_k - \kappa_l|^2}\hat{v}_{k,l})^2\} \leq [(\sum_{l:l\neq k} \frac{l^{-\beta}}{|\kappa_k - \kappa_l|^2} E\{\hat{v}_{k,l}^2\})^{1/2}]^2$$

$$\lesssim n^{-1}k^{-\alpha}(\sum_{l:l\neq k} \frac{l^{-\beta-\alpha/2}}{|\kappa_k - \kappa_l|^2})$$

bound $T_{j,k,2}$

Using the same splitting trick in the $T_{j,k,1}$ part, we can get

$$\sum_{l:l\neq k} \frac{l^{-\beta-\alpha/2}}{|\kappa_k-\kappa_l|^2} = \Big(\sum_{l=1}^{[k/C]} + \sum_{l=[k/C]+1}^{[Ck]} + \sum_{l=[Ck]+1}^{\infty}\Big)\frac{l^{-\beta-\alpha/2}}{|\kappa_k-\kappa_l|^2}$$

$$\lesssim \sum_{l=1}^{[k/C]} l^{3\alpha/2-\beta} + k^{2\alpha+2}\sum_{l=[k/C]+1}^{[Ck]} \frac{l^{-\beta-\alpha/2}}{|k-l|^2} +$$

$$k^{2\alpha}\sum_{l=[Ck]+1}^{\infty} l^{-\beta-\alpha/2}$$

$$\lesssim 1 + k^{3\alpha/2-\beta+1}\log k + k^{3\alpha/2-\beta+2} + k^{3\alpha-\beta+1}$$

$$\lesssim 1 + k^{3\alpha/2-\beta+2}$$

$$\Rightarrow |T_{j,k,2}| \lesssim n^{-1}k^{2\alpha-2\beta+4}$$

Summarize over $T_{j,k,1\sim4}$, (bound coefficient error)

$$\sum_{j=1}^{\infty}\sum_{k=1}^{m_n}(T_{j,k,1}^2+\ldots+T_{j,k,4}^2)=O_p(n^{-1}+n^{-2}\{m_n^3+m_n^{2\alpha-2\beta+5}(\log m_n)^2\})$$

$$=O_p(\frac{1}{n})$$

Bound the basis error

Using Parseval's identity,

$$\begin{aligned}
\iint \{\sum_{j=1}^{\infty}\sum_{k=1}^{m_n} b_{j,k}\{\phi_j \otimes (\hat{\phi}_k - \phi_k)\}\}^2 &= \sum_{j=1}^{\infty}\int \{\sum_{k=1}^{m_n} b_{j,k}(\hat{\phi}_k - \phi_k)\}^2 \\
&\lesssim m_n \sum_{j=1}^{\infty}\sum_{k=1}^{m_n} b_{j,k}^2 ||\hat{\phi}_k - \phi_k||^2 \\
&\lesssim m_n \sum_{k=1}^{m_n} k^{-2\beta} ||\hat{\phi}_k - \phi_k||^2 \\
&= O_p(n^{-1} m_n)
\end{aligned}$$

bound the high-order term

And the square of L2-norm of the high-order term is:

$$(\sum_{j=1}^{\infty}\sum_{k>m_n} b_{j,k}(\phi_j \otimes \phi_k))^2 = \sum_{j=1}^{\infty}\sum_{k>m_n} b_{j,k}^2 = O_p(m_n^{-2\beta+1})$$

(Orthogonality of the basis set)

Put all four terms together

$$|||\hat{b} - b|||^2 \lesssim \underbrace{\sum_{j=1}^{\infty}\sum_{k=1}^{m_n} \frac{1}{\hat{\kappa}_k^2}\left(\frac{1}{n}\sum_{i=1}^{n}\epsilon_{i,j}\hat{\xi}_{i,k}\right)^2}_{\text{eigenvalue error}=O_p(n^{-1}m_n^{\alpha+1}))} + \underbrace{\sum_{j=1}^{\infty}\sum_{k=1}^{m_n}(\dot{b}_{j,k} - b_{j,k})^2}_{\text{coefficient error}=O_p(n^{-1})}$$

$$+ \underbrace{\iint\{\sum_{j=1}^{\infty}\sum_{k=1}^{m_n} b_{j,k}\{\phi_j \otimes (\hat{\phi}_k - \phi_k)\}\}^2}_{\text{basis error}=O_p(n^{-1}m_n)} + \underbrace{\sum_{j=1}^{\infty}\sum_{k>m_n} b_{j,k}^2}_{\text{higher-order term}=O_p(m_n^{-2\beta+1})}$$

$$= O_p(n^{-1}m_n^{\alpha+1} + m_n^{-2\beta+1})$$

Take $m_n \sim n^{1/(\alpha+2\beta)}$, we have

$$|||\hat{b} - b|||^2 = O_p(n^{-\frac{2\beta-1}{\alpha+2\beta}})$$

Outline

Minimax rate

Cameron-Martin space

Let $P_{b,x}$ denote the distribution of $\int_I b(\cdot,t)x(t)dt + \varepsilon(\cdot)$, and P_0 denote the distribution of ε. Then the Cameron-Martin Space are defined as the following:

$$H = \{h = \sum_j h_j \phi_j : \sum_j \frac{h_j^2}{\lambda_j} < \infty\}$$

with the $< h, g >_H = \sum_j \frac{h_j g_j}{\lambda_j}, h = \sum_j h_j \phi_j, g = \sum_j g_j \phi_j$

Joint distribution of (X, Y)

Then the probability density can be formulated by Cameron-Martin formula:

$$p_{b,x}(y) = \frac{dP_{b,x}}{dP_0}(y) = \exp\{-\sum_j \frac{(\sum_k b_{j,k}x_k)^2}{2\lambda_j} + \sum_j \frac{y_j \sum_k b_{j,k}x_k}{\lambda_j}\}$$

where $y = \sum_j y_j \phi_j$.
If we denote by Q the distribution of X, then the joint distribution of (X, Y) is given by $p_{b,x}(y)dP_0(y)dQ(x)$

Estimation of distribution

Then let $\nu_n = [n^{1/(\alpha+2\beta)}]$, and

$$b^\theta = \sum_{\nu_n+1}^{2\nu_n} k^{-\beta}\theta_{k-\nu_n}(\phi_1 \otimes \phi_k)$$

where $\theta \in \{0,1\}^{\nu_n}$, thus the probability density function is the following:

$$p_{b^\theta,x}(y) = \exp\{-\frac{(\sum_{k=\nu_n+1}^{2\nu_n} k^{-\beta}\theta_{k-\nu_n}x_n))^2}{2\lambda_1} + \frac{y_1\sum_{k=\nu_n+1}^{2\nu_n} k^{-\beta}\theta_{k=\nu_n}x_k}{\lambda_1}\}$$

If we define $\tilde{p}_{\theta,x}(y) = p_{b^\theta,x}(y)$, then the estimated distribution is $\tilde{p}_{\theta,x}(y)dP_0(y)dQ(x)$ for each θ.
Let $(X_1, Y_1), ..., (X_n, Y_n)$ be i.i.d from $\tilde{P}_\theta$, then the estimator of b^θ is:

$$\bar{b}^n = \sum_{j,k} \bar{b}_{j,k}(\phi_j \otimes \phi_k)$$

Get the lower bound of $|||\bar{b}^n - b^\theta|||^2$

$\forall \theta, \theta' \in \{0,1\}^{\nu_n}$, let $\rho(\theta, \theta') = \sum_{k=1}^{\nu_n} |\theta_k - \theta'_k|$ (Hamming distance), then

$$P_\theta\{|||\bar{b}^n - b^\theta|||^2 \geq \frac{(2\nu_n)^{-2\beta}}{4} c\} \geq P_\theta\{\rho(\bar{\theta}^n, \theta) \geq c\}$$

According to Assouad's Lemma, we have

$$\max_\theta E_\theta\{\rho(\bar{\theta}^n, \theta)\} \geq \frac{\nu_n}{4} e^{-1/2\lambda_1}$$

Then apply Paley-Zygmund inequality, we have

$$P_\theta\{|||\bar{b}^n - b^\theta|||^2 \geq \frac{(2\nu_n)^{-2\beta}}{4} \frac{1}{16} e^{-1/(2\lambda_1)}\} \geq P_\theta\{\rho(\bar{\theta}^n, \theta) \geq \frac{\nu_n}{8} e^{-1/(2\lambda_1)}\}$$
$$\geq \frac{1}{16} e^{-1/2\lambda_1}$$

Get lower bound of $|||\bar{b}^n - b^\theta|||^2$

Therefore

$$\max_\theta P_\theta\{|||\bar{b}^n - b^\theta|||^2 \geq \frac{\mu_n^{-2\beta+1}}{2\beta+5} e^{-1/(2\lambda_1)}\} \geq \frac{1}{16} e^{-1/(2\lambda_1)}$$

Notice that $\nu_n^{-2\beta+1} \sim n^{-(2\beta-1)/(\alpha+2\beta)}$, then apply Chebyshev inequality, we have the lower bound like that in the lecture note:

$$\inf_{\bar{b}^n} \sup_\theta E_\theta |||\bar{b}^n - b^\theta|||^2 \gtrsim n^{-\frac{2\beta-1}{\alpha+2\beta}}$$

Summary

- Single Truncation estimator:

$$\hat{b}(s,t) = \sum_{k=1}^{m_n} \frac{1}{\hat{\kappa}_k} (\frac{1}{n} \sum \hat{\xi}_{i,k} Y_i(s)) \hat{\phi}_k(t)$$

- Covergence rate of ordinary least-square linear regression: $||\hat{b} - b||_2^2 \lesssim \sigma_p(X)^{-1} \sqrt{p/n}$.
- Convergence rate of PCA-based functional linear regression: $|||\hat{b} - b|||^2 \lesssim O_p(n^{-\frac{2\beta-1}{\alpha+2\beta}})$.

References I

M. Imaizumi, K. Kato

PCA-based estimation for functional linear regression with functional response.

arxiv:1609.00286v2, 2016.

K. Han, H. Shin

Functional linear regression for functional response via sparse basis selection

Journal of the Korean Statistical Society, 44, 376-389, 2015

J. Wang, J. Chiou, H. Mueller,

Review of Functional Data Analysis,

arXiv:1507.05135, 2015

J. Morris,

Functional Regression,

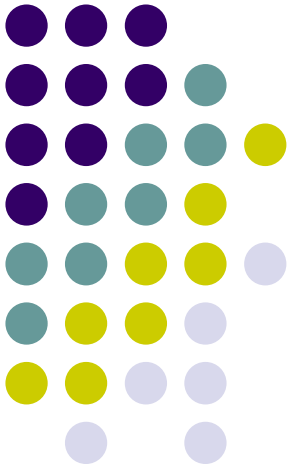
arXiv:1406.4068, 2014

References II

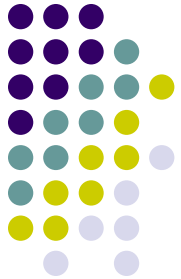

J. Ramsay, B. Silverman,
Functional Data Analysis,
Springer Series in Statistics Chapter 16, 2005

Inference of Sparse Gaussian Graphical Models: Algorithms and Theory

Ifigeneia Apostolopoulou



Applications of Gaussian Graphical Models (GGM)



- Popular tool for learning network structure over a large number of continuous variables
  - Neuroscience
  - Computational Biology
  - Natural Language Processing
  - Computational Finance
  - Energy Forecasting

Mathematical Formulation

Given $X_1, X_2, \ldots, X_N \sim N(\mu, \Sigma), X_i \epsilon R^p$, estimate the covariance matrix $\Omega = \Sigma^{-1}$

The solution of this problems leads to inference of the undirected graphical model since for $W = (W_1, \ldots, W_p) \sim N(\mu, \Sigma)$:

$$W_i \perp W_j || (W_k, k \neq j, k \neq i) \Leftrightarrow \Omega_{ij} = 0$$

Proposed Approaches

- Graphical Lasso :

$$\max_{\boldsymbol{\Omega}>0} \log(\det(\boldsymbol{\Omega})) - tr(\boldsymbol{\Omega}\boldsymbol{S}) - \rho\|\boldsymbol{\Omega}\|_1$$

, where $\boldsymbol{S}$ the sample covariance matrix

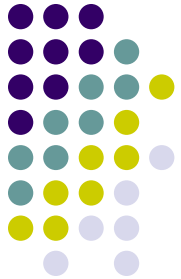
- In the high-dimensional case: $p > n$

$\min\|\boldsymbol{\Omega}\|_1$ subject to $|\boldsymbol{S}\boldsymbol{\Omega} - \boldsymbol{I}|_\infty \leq \lambda$

Estimators offer convergence rates $O_P(s\sqrt{logp/n})$

if the true precision matrix is s −sparse (at most s non zero entries per row)

Conditional Gaussian Graphical Models



- Ignore correlations between input variables
- $\boldsymbol{X_1, X_2, \ldots, X_N, X_i \epsilon R^p}$ the input variables, $\boldsymbol{Y_1, Y_2, \ldots, Y_N, Y_i \epsilon R^q}$ the output variables learn $\boldsymbol{\Omega_{xy}, \Omega_{yy}}$ such that

$$p(\boldsymbol{y}|\boldsymbol{x}) \sim N(-\boldsymbol{\Omega_{yy}^{-1}\Omega_{xy}^{T}x},\ \boldsymbol{\Omega_{yy}^{-1}})$$

- This is a more detailed formulation of the multiple regression:
  $\boldsymbol{y = Bx + \epsilon}$

Proposed Approaches

- Solution of the optimization problem:

$$\min_{\Omega_{yy}\succeq 0,\Omega_{xy}\succeq 0} -log(det(\Omega_{yy})) + tr(\mathrm{S}_{yy}\Omega_{yy} + 2\mathrm{S}_{xy}^{\mathrm{T}}\Omega_{xy} + \Omega_{yy}^{-1}\Omega_{xy}^{\mathrm{T}}\mathrm{S}_{xx}\Omega_{xy}) + \lambda_1||\Omega_{xy}||_1 + \lambda_2||\Omega_{yy}||_1$$

- Second-order coordinate descent methods

Subspace Clustering

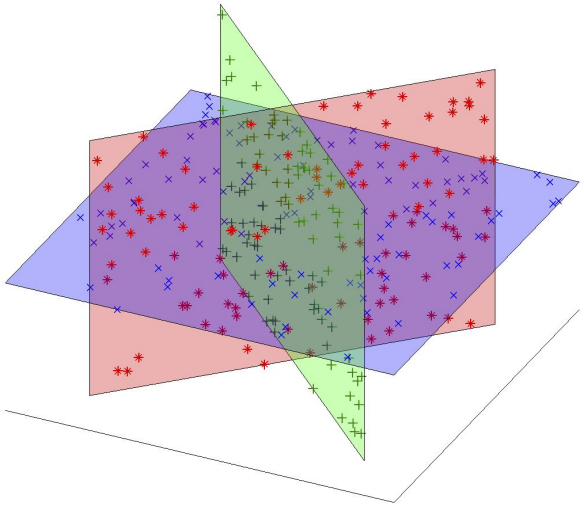
Alan Mishler & Deepika Bablani

Problem: Clustering is challenging in high dimensions

- Concentration of distance
- Clusters may exist in different lower-dimensional spaces

Solution: Look for clusters in lower dimensional subspaces

- Bottom-up algorithms
- Top-down algorithms
- Model-based methods
- Sparse subspace clustering



Bottom-up algorithms (SUBCLU)

$X_i \in \mathbb{R}^d, i = 1, \ldots, n$

- Builds on density based clustering algorithm DBSCAN.
- Can find clusters in axis parallel subspaces.
- Utilizes downward closure property: if cluster is found in subspace S, then each subspace T of S also contains a cluster. However, a cluster C in subspace C $\subseteq$ DB not necessarily a cluster in T since clusters are required to be maximal, and more objects might be contained in the cluster T that contains C. However, a density connected set in S is also a density-connected set in T.

Top-down algorithms (PROCLUS, ORCLUS)

$$X_i \in \mathbb{R}^d, i = 1, \ldots, n$$

PROCLUS (“PROJected CLUStering”)

- Goal:
  - Partition the data into clusters C1, C2, …, Ck, plus a set of outliers
  - Partition the features into sets of dimensions D1, D2, …, Dk corresponding to each cluster
- Method: select k medoids, iteratively assign data points to clusters and reduce dimensionality of each cluster

ORCLUS (“abitrarily ORiented projected CLUSter generation”)

- Extension of PROCLUS, looks for non-axis parallel subspaces.

Model-based methods (EPGMMs) $X_i \in \mathbb{R}^d, i = 1, \ldots, n$

Treat each data point as a linear combination of normally-distributed latent factors, plus normal noise. If X_i is in cluster g, then:

$$X_i = \mu_g + \Lambda_g U_{ig} + \epsilon_i$$

$\Lambda_g : d \times q$ matrix of factor weights

$$U_{ig} \sim N(0, I_q)$$

$$\epsilon_i \sim N(0, \Sigma)$$

Estimate parameters using Alternating Expectation Conditional Maximization (AECM) algorithm

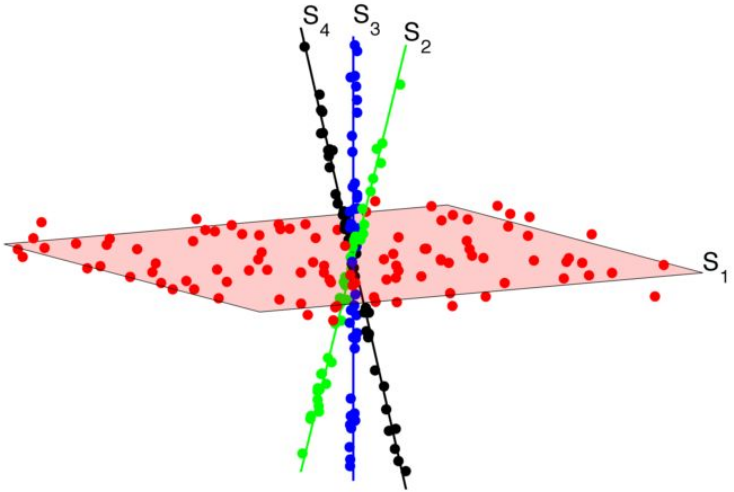
Sparse subspace clustering

$$X_i \in \mathbb{R}^d, i = 1, \ldots, n$$

Goal: Represent each data point as a sparse combination of other data points.

Assume data come from a union of linear subspaces. The best sparse representation of each data point should only involve data points in the same subspace.

Advantages: no need to specify number of clusters or dimensions of subspaces in advance.



Conclusions

- Many different approaches
- Since there’s no universal definition of cluster or a subspace, there’s no “best” algorithm
- Theoretical performance guarantees exist only for sparse subspace clustering

Comparison of Dantizig Selector and Lasso

Xiaoyi Gu, Yufei Yi

May 2, 2017

Formulation

Goal

Under $y \in \mathbb{R}^n$, $X \in \mathbb{R}^{n \times p}$ *with* $n \ll p$, *and* $z \sim \mathcal{N}(0, \sigma^2 I)$ *noise, want to estimate* $\beta \in \mathbb{R}^p$ *from*

$$y = X\beta_0 + z, \tag{1}$$

Lasso

$$\min_{\beta} \|y - X\beta\|_{\ell_2} + \lambda \|\beta\|_{\ell_1}. \tag{2}$$

Dantizig Selector

$$\min_{\beta} \|\beta\|_{\ell_1} \text{ subject to } \|X^*(y - X\beta)\|_{\ell_\infty} \leq \lambda_p \sigma. \tag{3}$$

Three Major Assumptions on X

UUP(Uniform Uncertainty Principle)

$\exists\ S < p$ *such that* $\forall |T| \leq S$, $\exists \delta$ *such that for all* $c \in \mathbb{R}^{|T|}$*:*

$$(1-\delta)\|c\|_{\ell_2}^2 \leq \|X_T c\|_{\ell_2}^2 \leq (1+\delta)\|c\|_{\ell_2}^2$$

IDC(Incoherent Design Condition)

For β_0 *being* S_n*-sparse with* $\lim_{n\to\infty} S_n = \infty$*.* $\exists e_n > 0$ *such that:*

$$\liminf_{n\to\infty} \frac{e_n \phi_{min}(e_n^2 S_n)}{\phi_{max}(S_n+n)} \geq 18$$

MIC(Mutual Incoherent Condition)

$\rho(S) = \max\{|\langle X_i, X_j\rangle| : i \in T, j \in T^c, |T| \leq S\}$*.* X *satisfies MIC if* $\rho(S)S \leq 1/K$ *for some* $K > 0$*.*

Comparison under the Restricted Eigenvalue Condition

REC(S, S', C_0) Restricted Eigenvalue Condition

$$\kappa(S, S', C_0) := \min_{T:|T|\le S} \min_{c:\|c_{T^c}\|_{\ell_1} \le C_0 \|c_{T\cup R}\|_{\ell_1}} \frac{\|Xc\|_{\ell_2}}{\sqrt{n}\|c_T\|_{\ell_1}} > 0,$$

where R corresponds to the S' coordinates of $|c|$ outside T.

Theorem [Bickel, Ritov,Tsybakov]

Suppose β_0 is S-sparse and all diagonal elements of $1/n(X^*X)$ is 1,

- *Under RE$(S, S', 1)$ and $\lambda_p = \sigma\sqrt{\log p/n}$,*

$$\|\hat{\beta}_d - \beta_0\|_{\ell_2}^2 \le CS(1 + \sqrt{S/S'})^2 \frac{\sigma^2 \log p}{n\kappa^4(S, S', 1)}$$

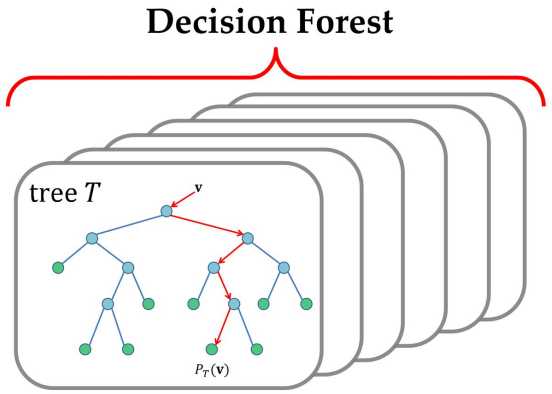
- *Under RE$(S, S', 3)$ and $\lambda = \sigma\sqrt{n \cdot \log p}$,*

$$\|\hat{\beta}_l - \beta_0\|_{\ell_2}^2 \le C'S(1 + 3\sqrt{S/S'})^2 \frac{\sigma^2 \log p}{n\kappa^4(S, S', 3)}$$

Statistical Analysis of Random Forests

Ritesh Noothigattu, Ben Parr

Breiman (2001)



- Formally defines a random forest

- Main results
  - No overfitting as more trees are added
  - Error depends on:
    - Individual tree strength
    - Correlation between trees

$$PE^* \leq \bar{\rho}(1-s^2)/s^2$$

Biau (2012), Denil et al (2014)

- Biau (2012)
  - Consistency on a previously proposed variant
  - Convergence rate depends only on number of strong features

- Denil et al (2014)
  - A new theoretically tractable variant
  - Proves consistency

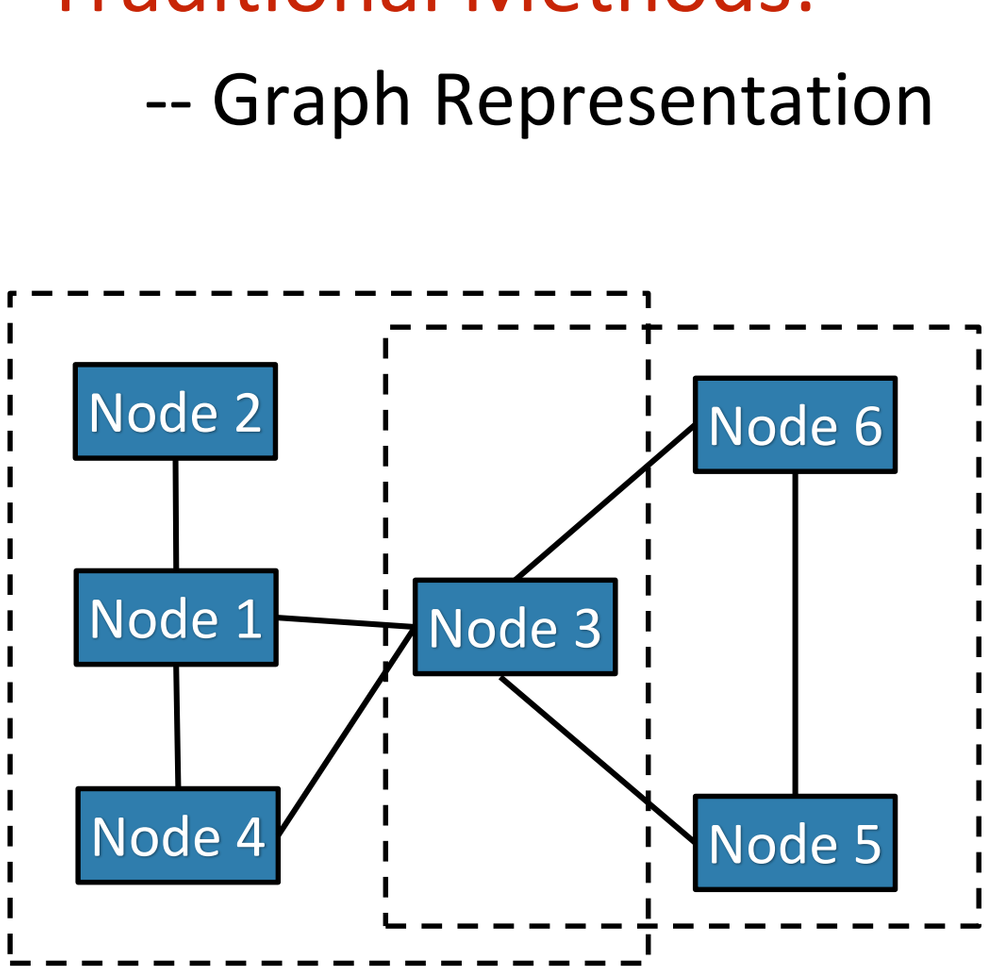
$$\mathbb{E}[r_n(\mathbf{X}) - r(\mathbf{X})]^2 \to 0 \text{ as } n \to \infty$$

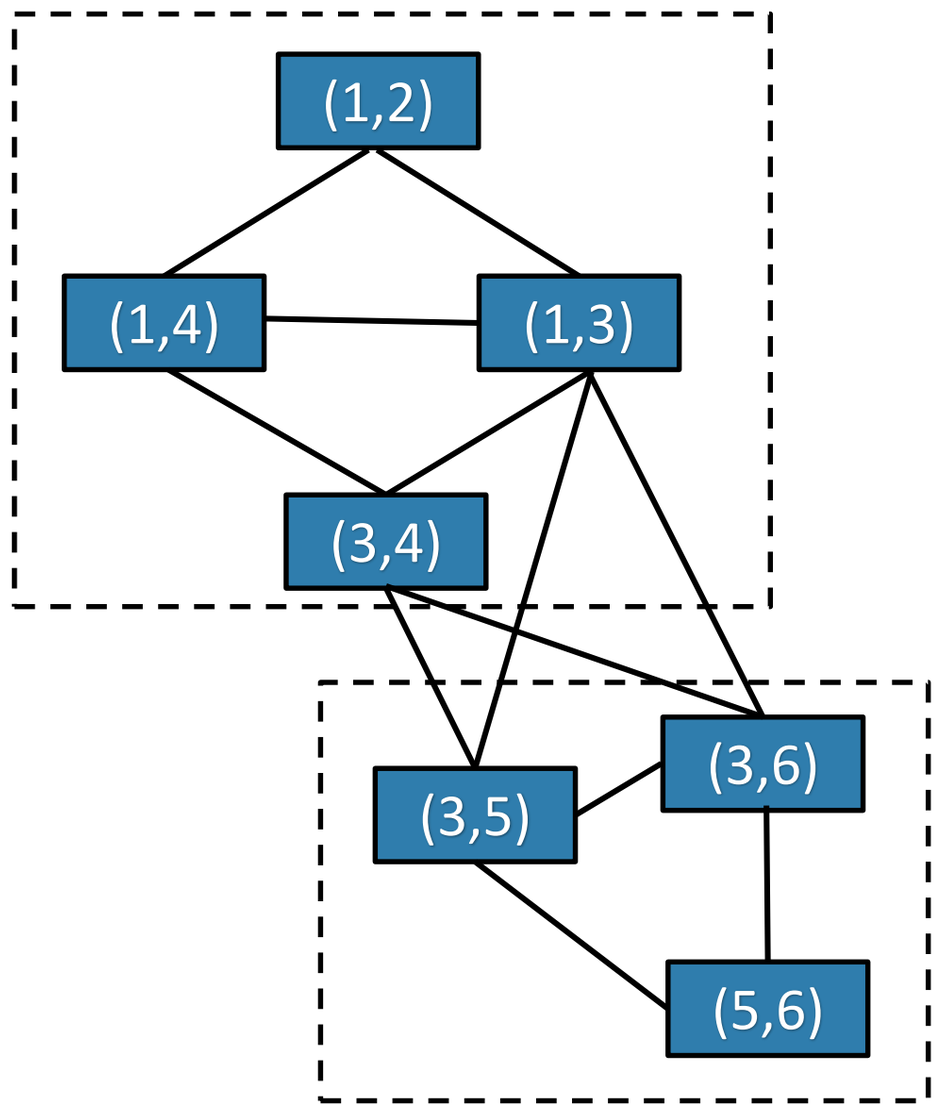
Mix-membership Clustering

Boyan Duan, Xiaoyi Yang

Traditional Methods:

-- Graph Representation





Traditional Methods:

-- Extend Traditional K-means (NEO K-Means)

1. Traditional K-means Assignment matrix Y_{ij}
   Constrain: $trace(U^T U) = n$, Row sum of Y is **1** vector.

2. NEO K-means Assignment matrix U_{ij}
   Constrain: $trace(U^T U) = (1+\alpha)n$, Row sum of $U \leq \beta n$.

3. Replace Y in the objective function of K-means into U.

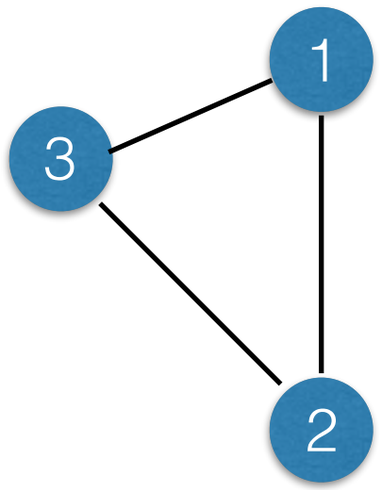
$$min_U \sum_{j=1}^{k}\sum_{i=1}^{n} u_{ij} w_i ||\phi(x_i) - m_j||^2 = min_U \sum_{j=1}^{k}(\sum_{i=1}^{n} u_{ij} w_i K_{ii} - \frac{u_j^T WKW u^j}{u_j^T W u_j})$$

where $m_j = \frac{\sum_{i=1}^{n} u_{ij} w_i \phi(x_i)}{\sum_{i=1}^{n} u_{ij} w_i}$ s.t. $trace(U^T U) = (1+\alpha)n$, $\sum_{i=1}^{n} \mathbb{1}\{(U1)_i = 0\} \leq \beta n$

ϕ is the non-linear mapping, W is the diagonal weight matrix and K is the kernal matrix.

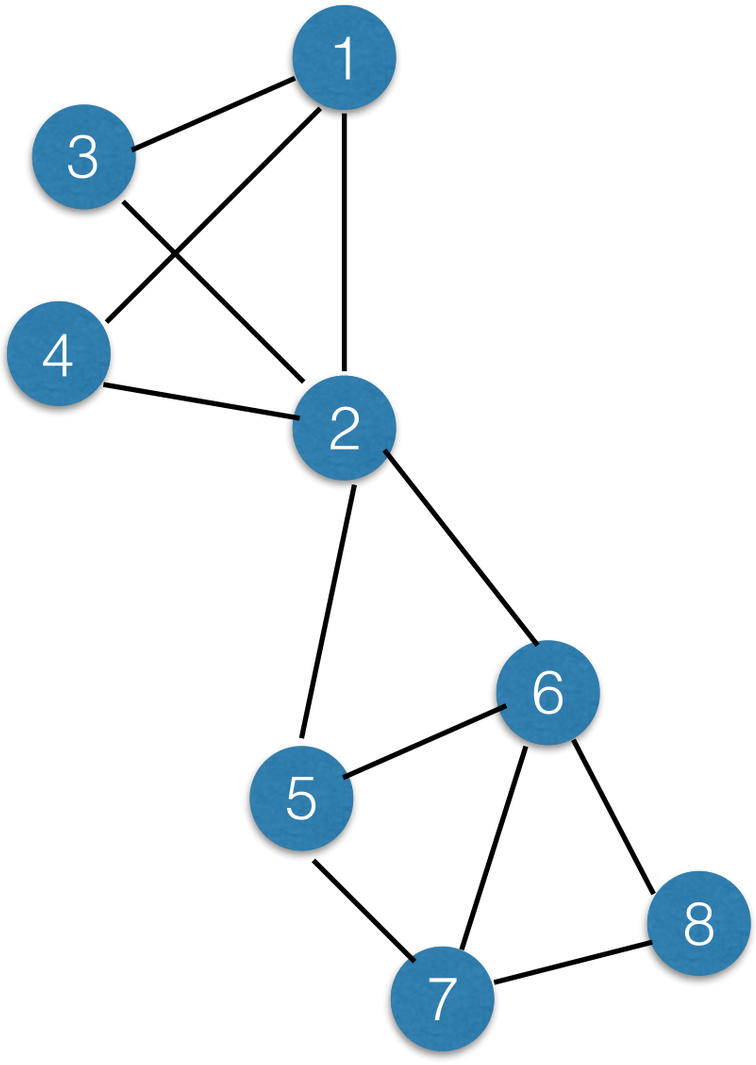
Clustering based on motif network

Motif network



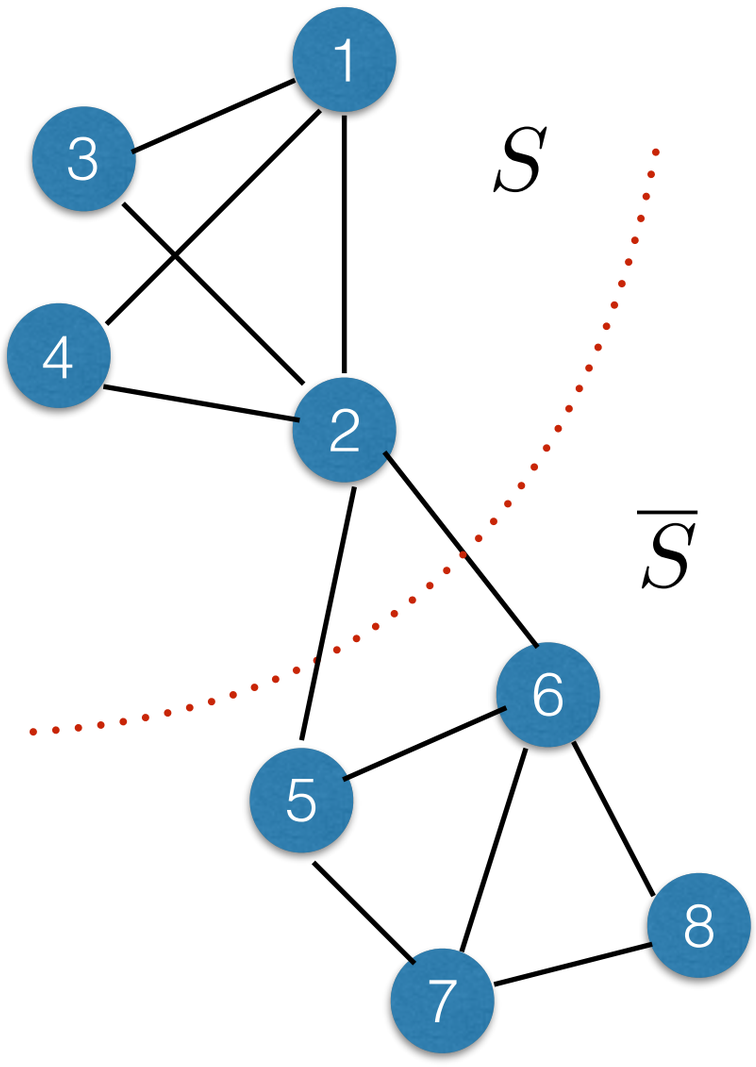
Clustering based on motif network

Motif network



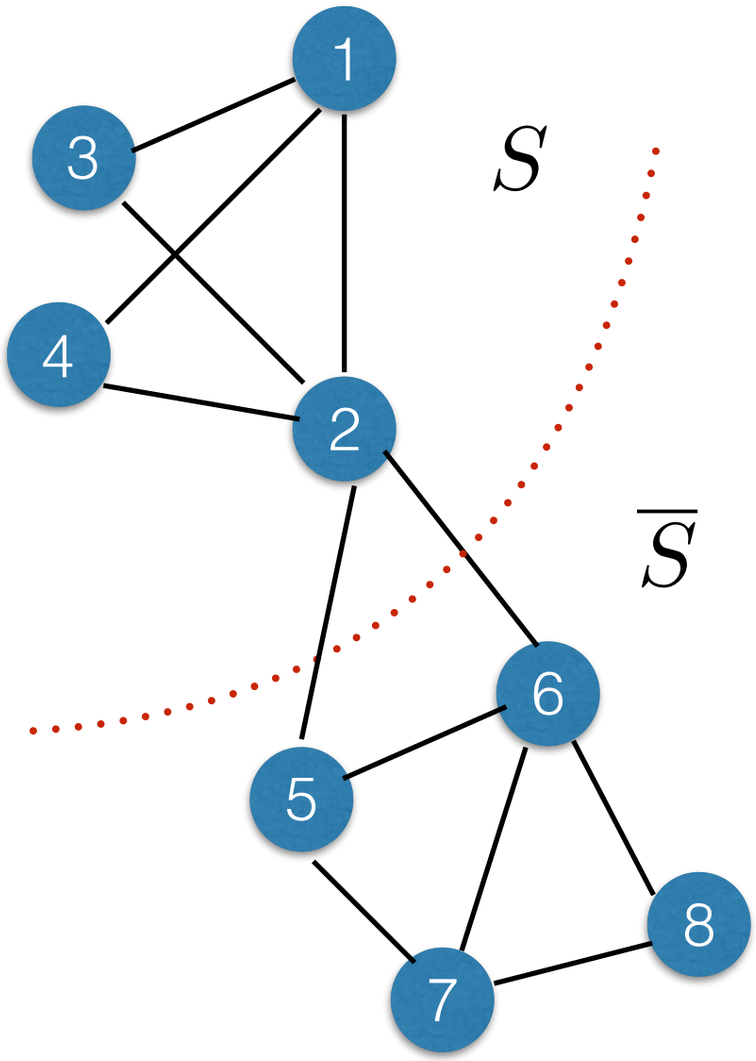
Clustering based on motif network

Motif network



Clustering based on motif network

Motif network



Objective:

$$min_{S_1,\dots,S_k} \sum_{j=1}^{k} \frac{cut(S_j, \overline{S_j})}{min(vol(S_j), vol(\overline{S_j}))}$$

$$s.t. \sum_{j=1}^{k} \#\{nodes \in S_j\} \leq (1+\alpha)n$$

degree of overlapping

Mathematical guarantee:

Spectral graph methodology for weighted graph

Thank you!

Efficient PAC Reinforcement Learning in Contextual Decision Processes

Karan Goel
Deepak Dilipkumar

Problem Statement

- PAC Reinforcement Learning: Learn a near-optimal policy with high probability in sample efficient way

- Reinforcement Learning v/s Supervised Learning: Samples not iid

Contextual Decision Process

- A recent general framework to model the world in Krishnamurthy et al. (2016)
- Subsumes Markov Decision Processes, Partially Observable Markov Decision Processes

Bellman Rank of a CDP

- New measure by Jiang et al. (2016) that characterizes the complexity of a CDP
- Most practical problems actually have low Bellman rank

Main Paper

- Jiang et al. (2016): Can learn policy for CDP with low Bellman Rank with high sample efficiency
- Algorithm that outputs a near optimal policy with high probability given CDP with low Bellman Rank

Thanks!

Online Non-stationary Time Series Regression with Autoregressive Models

Lisheng Gao, Rui Peng

Motivation

- Problem Definition

- Time Series: sequence of observations indexed in order (normally chronological)

$$x_t = f(t) + u_t \quad t = 1, ..., n$$

- Stationarity: finite variation, constant first moment and second moment across time
- Online Optimization: iterative parameter tuning, incomplete knowledge of the future

- Why Important?

- Natural temporal ordering
- Extremely wide applications (signal processing, weather forecasting, mathematical finance…)

- Connection with this course

Preliminaries

- Autoregressive Models:

- ARMA (Autoregressive moving average)

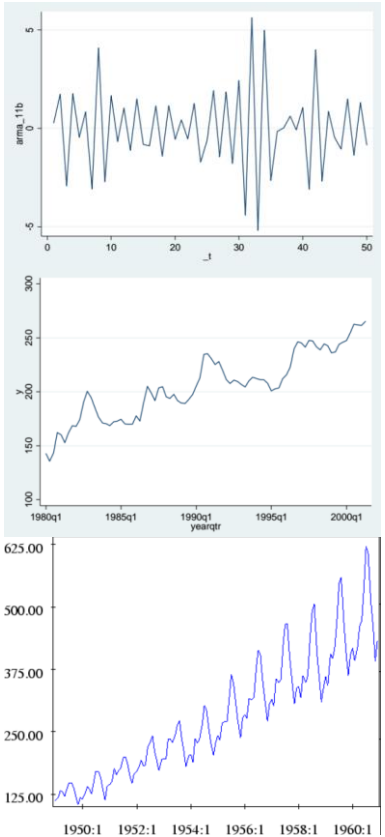
$$ARMA(p,q): \quad X_t = \mu + \sum_{i=1}^{p} \beta_i X_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t$$

- ARIMA (Autoregressive integrated moving average)

$$ARIMA(p,d,q): \quad (1 - \sum_{i=1}^{p} \beta_i B^i)(1-B)^d X_t = \mu + (1 + \sum_{i=1}^{p} \theta_i B^i)\epsilon_t$$

- SARIMA

$$SARIMA(p,d,q) \times (P,D,Q)_s: \quad \beta(B)\Phi(B^s)\Delta^d \Delta_s^d X_t = \theta(B)\Theta(B^s)\epsilon_t$$



Theories – ARMA online gradient descent [1]

- Loss – $f_t(\alpha, \beta) = l_t\left(X_t, \tilde{X}_t(\alpha, \beta)\right) = l_t\left(X_t, \left(\sum_{i=1}^{p} \alpha_i X_{t-i} + \sum_{i=1}^{q} \beta_i \epsilon_{t-i}\right)\right)$
- Regret – $R_T = \sum_{t=1}^{T} l_t(X_t, \tilde{X}_t) - \min_{\alpha,\beta} \sum_{t=1}^{T} l_t\left(X_t, \tilde{X}_t(\alpha, \beta)\right)$

Algorithm 1: ARMA-OGD(k,q)

Input: ARMA order k, q; learning rate η; an initial $(m+k) \times (m+k)$ matrix A_0.
Set $m = q \cdot \log_{1-\epsilon} \frac{1}{T \cdot L \cdot M_{max}}$.
Choose $\gamma^1 \in \mathcal{K}$ arbitrarily.
for $t = 1$ *to* $T-1$ **do**
  Predict $\tilde{X}_t(\gamma) = \sum_{i=1}^{m+k} \gamma_i X_{t-i}$.
  Observe X_t and suffer loss $l_t^m(\gamma^t)$.
  Let $\nabla_t = \nabla l_t^m(\gamma^t)$
  Set $\gamma^{t+1} \leftarrow \Pi_{\mathcal{K}}^{A_t}\left(\gamma^t - \frac{1}{\eta}\nabla_t\right)$.
end

Theories – generalized online gradient descent [2]

- Loss – $l_t^M(\gamma) = l_t\left(x_t, \zeta(\tau(\tilde{x_t}))\right) = l_t\left(x_t\zeta\left(\sum_{i=1}^{M}\gamma_i\tau(x_{t-i})\right)\right)$

	$\tau(x_t)$	$\zeta(\tilde{x_t})$
ARMA	x_t	$\tilde{x_t}$
ARIMA	$\Delta^d x_t$	$\tilde{x_t} + \sum_{i=0}^{d-1}\Delta^i x_{t-1}$
SARIMA	$\Delta^d\Delta_s^D x_t$	$\tilde{x_t} + \sum_{i=0}^{d-1}\Delta^i\Delta_s^D x_{t-1} + \sum_{i=0}^{D-1}\Delta_s^i x_{t-s}$

Algorithm 2: TSP-OGD Framework

Input: Model parameters l_a, l_m; Horizon T; Learning rate η; Data $\{x_t\}$; Transformation τ; Inverse Transformation ζ.

Set $m = \log_{\lambda_{max}}(2\kappa TLM_{max}\sqrt{l_m})^{-1} + l_a$.

Choose $\gamma^1 \in \mathcal{K}$ arbitrarily.

for $t = 1$ *to* $T - 1$ **do**

Transform x_t to get $\tau(x_t)$ $\tau(\tilde{x_t}) = \sum_{i=1}^{m}\gamma_i\tau(x_{t-i})$ Predict $\tilde{X}_t(\gamma) = \zeta(\tau(\tilde{x_t}))$.

Observe X_t and receive loss $l_t^m(\gamma^t)$.

Set $\gamma^{t+1} \leftarrow \Pi_{\mathcal{K}}\left(\gamma^t - \frac{1}{\eta}\nabla l_t^m(\gamma^t)\right)$.

end

Theory – theoretical bound

Theorem 3.2 *Let* $\eta = \frac{D}{G\sqrt{T}}$. *For any data sequence* $\{x_t\}_{t=1}^{T}$ *that satisfies the assumptions, Algorithm* 2 *generates a sequence* $\{\gamma^t\}$ *in which*

$$\sum_{t=1}^{T} l_t^m(\gamma^t) - \min_{\alpha,\beta\in\mathcal{E}} \sum_{t=1}^{T} E[f_t(\alpha,\beta)] = O\left(DG\sqrt{T}\right) \tag{18}$$

References

- [1] Oren Anava, Elad Hazan, Shie Mannor, and Ohad Shamir. Online learning for time series prediction. arXiv preprint arXiv:1302.6927, 2013.
- [2] Christopher Xie, Avleen Bijral, and Juan Lavista Ferres. An online prediction framework for non-stationary time series. arXiv preprint arXiv:1611.02365, 2016.

Learning Sequential Data: Hidden Markov Models and Beyond

Siqi Chen

10702 Project Presentation

CMU

Motivation

- For many application such as speech recognition, bioinformatics musical score following and stock prediction, the time series nature of the data is intrinsic, so the usual i.i.d. assumption is no longer appropriate
- HMM is a prototypical methods for modeling times series
- However, it faces several important restrictions:

1. The number of hidden states is finite

2. If we have large number of possible states, then standard HMM may require learning too many parameters.

3. Maximum likelihood estimation procedures do not consider the complexity of the model, making it hard to avoid over or underfitting

Motivation

- So we consider the following extensions of HMM:
- Factorial HMM
- Switiching State Space Models
- Infinite Hidden Markov Model

- To effectively discuss iHMM, need to consider Hierarchical Dirichlet Process

- This brings the problem of inference algorithms:
- Variational Learning for Switching State Space Models
- Beam Sampling for Infinite Hidden Markov Models

Connection with class

- An in-depth study of some particular instances (HMM and its extensions) of dynamic Bayesian network and graphical model
- Hierarchical Dirichlet process (HDP) can be used for clustering
- By placing HMM in the general graphical model framework, the extensions and inference algorithms become natural

Literature

- Matthew J. Beal, Zoubin Ghahramani, and C. E. Rasmussen. The infinite hidden markov model.Neural Information Processing Systems, 14:577–585, 2002.
- Z. Ghahramani and G. E. Hinton. Variational learning for switching state-space models. Neural Computation, 12(4):831–864, April 2000. ISSN 0899-7667.
- Zoubin Ghahramani. An introduction to hidden markov models and bayesian networks. International Journal of Pattern Recognition and Artificial Intelligence, 15:9 – 42, 2001.
- Michael Jordan. Graphical models. Statistical Science, Vol. 19:140 – 155, 2004.
- Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical dirichlet processes.Journal of the American Statistical Association, 101(476):1566–1581, 2006.
- Jurgen Van Gael, Yunus Saatci, Yee Whye Teh, and Zoubin Ghahramani. Beam sampling for the

infinite hidden markov model. In Proceedings of the 25th international conference on Machine learning, pages 1088–1095. ACM, 2008.

HMM

- (i) Y_t is generated by a process whose state S_t is hidden from the observer.
- (ii) S_t has the Markov property: E[f(S_t)|F_s] = E[f(S_t)|S_s]
- (iii) S_t is discrete: S_t can take K values {1,…,K}
- The model:

$$P(S_1, \ldots, S_T, Y_1, \ldots, Y_T) = P(S_1)P(Y_1|S_1)\Pi_{t=2}^{T} P(S_t|S_{t-1})P(Y_t|S_t) \quad (1)$$

which means to specify the joint distribution, we only need to specify the initial distribution $P(S_1)$, the $K \times K$ time invariant transition matrix $P(S_t|S_{t-1})$ and the $K \times L$ time invariant emission matrix $P(Y_t|S_t)$, assuming Y_t only takes L discrete values. We can further augment the HMM by assuming we have additional input variables U_t, and that the state transition probability is input dependent $P(S_t|S_{t-1}, U_t)$ Ghahramani [2001].

Factorial HMM

We represent the state variables by a collection of discrete state varaibles: $S_t = S_t^{(1)}, ..., S_t^{(M)}$, each $S_t^{(m)}$ can take $K^{(m)}$ values. For simplicity, let $K^{(m)} = K, \forall m$. So the state space has K^M possible values. Letting each state variable evolve according to its own dynamcis, we specify the following model:

$$P(S_t|S_{t-1}) = \Pi_{i=1}^{M} P(S_t^{(i)}|S_{t-1}^{(i)}) \tag{4}$$

So there are only M $K \times K$ transition matrices to estimate. The observation Y_t depends on all state varaibles $S_t^{(i)}, ..., S_t^{(M)}$. One particular choice is letting Y_t be a Gaussian random vector whose mean is a linear function of the state varaibles.

$$P(Y_t|S_t) = |R|^{-1/2}(2\pi)^{-D/2} \exp\{-\frac{1}{2}(Y_t - u_t)^T R^{-1}(Y_t - u_t)\} \tag{5}$$

with

$$u_t = \sum_{i=1}^{M} W^{(i)} S_t^{(i)} \tag{6}$$

where $W^{(m)}$ is a $D \times K$ matrix and R is the $D \times D$ covariance matrix. From (3) we see that there are K^M possible mean vectors, so this is a Gaussian mixture model with K^M components, each of which having covariance matrix R.

Factorial HMM

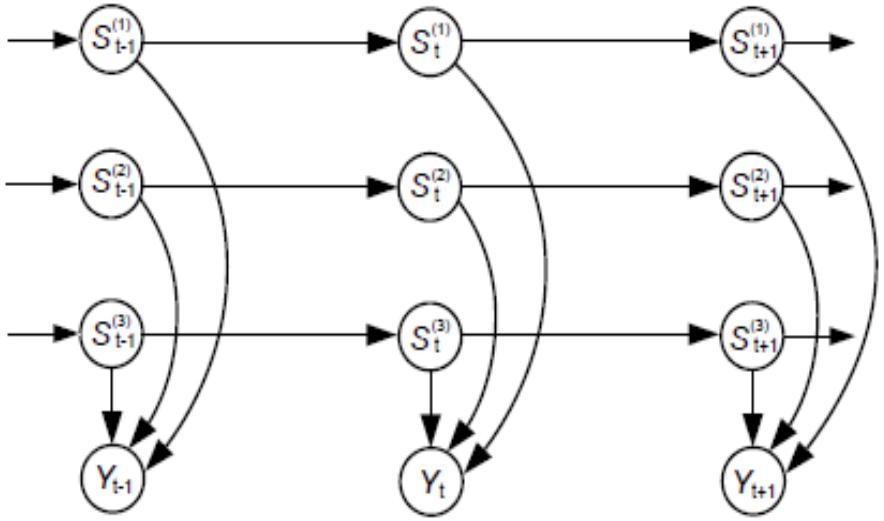


Figure 5: A Bayesian network representing the conditional independence relations in a factorial HMM with $M = 3$ underlying Markov chains. (We only show here a portion of the Bayesian network around time slice t.)

Switching State Space Models

We now intorduce the extension that combine continuous and discrete state spaces. Let $Y_{1:T}$ be the sequence of observations, $X_{1:T}^{(1)}, ..., X_{1:T}^{(M)}$ be M real valued state vectors, S_t be one discrete state varaibles. The model is specified as:

$$P(S_{1:T}, X_{1:T}^{(1)}, ..., X_{1:T}^{(M)}, Y_{1:T}) = P(S_1)\Pi_{t=2}^{T} P(S_t|S_{t-1})\Pi_{m=1}^{M}[P(X_1^{(m)})\Pi_{t=2}^{T} P(X_t^{(m)}|X_{t-1}^{(m)})]$$
$$\times \Pi_{t=1}^{T} P(Y_t|X_t^{(1)}, ..., X_t^{(M)}, S_t)$$

Conditioned on a switch state $S_t = m$, the observable is a multivaraite Gaussian with output equation given by the state space model m. I.e.,

$$P(Y_t|, X_t^{(1)}, ..., X_t^{(M)}, S_t = m) = (2\pi)^{-D/2}|R|^{-1/2}$$
$$\times \exp\{-\frac{1}{2}(Y_t - C^{(m)}X_t^{(m)})^T R^{-1}(Y_t - C^{(m)}X_t^{(m)})\}$$

where R is the covariance matrix of Y_t, $C^{(m)}$ is the output matrix for state space model m, as in (3). In this model, the switch variable is itself a discrete Markov chain with initial probability $P(S_t)$ and transition matrix $P(S_t|S_{t-1})$. It "gates" the output of M state space models. And each real valued state vectors $X_{1:T}^{(m)}$ follows a different state space model as in (2).

DP, HDP

In order to effectively discuss Infinite Markov Model, we first introduce Dirichlet Process and Hierarchical Dirichlet Process.

The Dirichlet process is a stochastic process used in Bayesian nonparametric models of data. Let H be a distribution over Θ and α be a positive real number. G is Dirichlet Process with base distribution H and concentration parameter α, $G \sim DP(\alpha, H)$, if

$$(G(A_1), ..., G(A_r)) \sim \text{Dirichlet}(\alpha H(A_1), ..., \alpha H(A_r)) \tag{7}$$

for every finite measurable partition $A_1, ..., A_r$ of Θ. The base distribution H is the mean of the DP: for any measurable set $(\forall A \subset \Theta) E[G(A)] = H(A)$. On the other hand, the concentration parameter α can be understood as an inverse variance: $V[G(A)] = H(A)(1 - H(A))/(\alpha + 1)$. α is also called the strength parameter, since when we use the DP as a nonparametric prior over distributions in a Bayesian nonparametric model, it can be interpreted as the strength of the prior.

A hierarchical Dirichlet process (HDP) is a set of Dirichlet processes (DPs) coupled through a shared random base measure which is itself drawn from a DP[Teh et al., 2006]. In other words, each $G_k \sim DP(\alpha; G_0)$ with shared base measure G_0 and the shared base measure G_0 is itself given a DP prior: $G_0 \sim DP(\gamma, H)$, with H as the global base measure.

iHMM

- The Infinite Hidden Markov Model (iHMM), is a non-parametric Bayesian extension of the Hidden Markov Models with a infinite number of hidden states.
- While in principle this will require the state transition matrix have infinite number of parameters to estimate, the theory of Dirichlet Processes (DPs) enables us to implicitly integrate the parameters out, leaving only a few hyperparameters defining the prior
- HMM involves not a single mixture model, but rather a set of mixture models—one for each value of the current state.
- The current state S_t indexes a specific row of the transition matrix, with the probabilities in this row serving as the mixing proportions for the choice of the next state S_{t+1}
- Thus, to consider a nonparametric variant of the HMM that allows a countable number of states, we must consider a set of DPs, one for each value of the current state. Moreover, these DPs must be linked, because we want the same set of next states to be reachable from each of the current states

iHMM

In particular, we model each row of the transition matrix and emission matrix of HMM as a DP. By identifying each G_k in the HDP described above with both the transition probabilities $\pi_{k,k'}$ from state k to k' and the emission distributions parametrized by $\phi(k')$, i.e. $Y_t|S_t \sim F(\phi_{S_t})$, a Infinite Hidden Markov Model (HDP-HMM) is:

$$\beta \sim \text{GEM}(\gamma), \pi_k|\alpha, \beta \sim \text{DP}(\alpha, \beta), \phi_k \sim H \tag{8}$$

$$s_t|s_{t-1} \sim \text{Multinomial}(\pi_{s_{t-1}}), Y_t|S_t \sim F(\phi_{S_t}) \tag{9}$$

iHMM

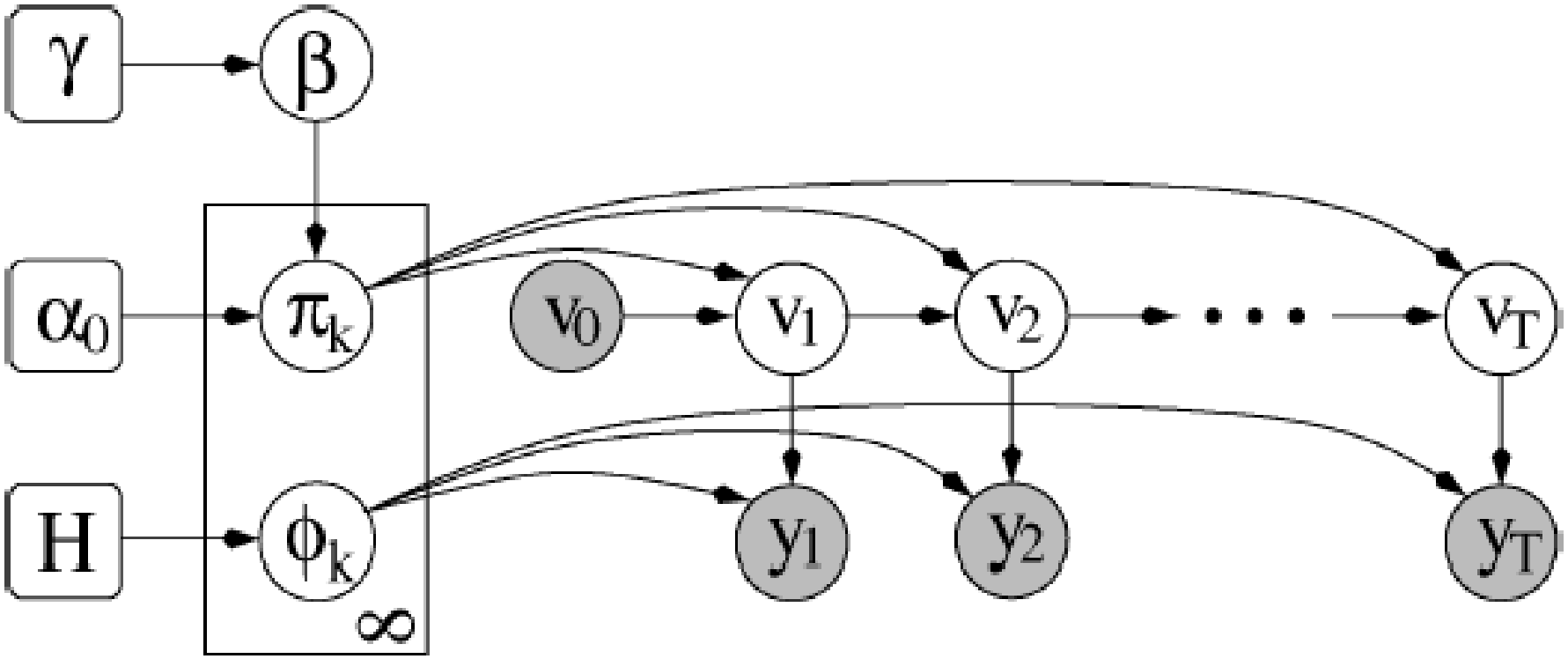


Figure 6. A Graphical Representation of an HDP–HMM.

Beam Sampling: why?

- iHMM as a non-parametric Bayesian extension of the HMM.
- In general, non-parametric Bayesian models are models of infinite capacity, a finite portion of which is used to model a finite amount of data. The usual idea of searching/averaging over the space of finite models is replaced with Bayesian inference over the size of submodel used to explain the data.
- exact Bayesian inference for the iHMM is intractable. The usual forwardbackward algorithm cannot be applied since the number of states are infinite.

Beam Sampling: idea

- Beam sampling combines the idea of slice sampling and dynamic programming.
- The idea of beam sampling is to introduce auxiliary variables u such that conditioned on u the number of trajectories with positive probability is finite.
- We then apply dynamic programming to compute the conditional probabilities of these trajectories and thus sample whole trajectories efficiently.
- Note that the marginal distribution of other variables is not changed by the introduced auxiliary variable U.

Thank you!

Theoretical basis for neural networks training techniques

Igor Gitman
igitman@andrew.cmu.edu

Key idea: natural gradient descent (NGD)

Gradient descent: $\theta_{k+1} = \theta_k - \alpha_k \nabla f(\theta_k)$

Key idea: natural gradient descent (NGD)

Gradient descent: $\theta_{k+1} = \theta_k - \alpha_k \nabla f(\theta_k)$

Newton method: $\theta_{k+1} = \theta_k - \alpha_k \left[\nabla^2 f(\theta_k)\right]^{-1} \nabla f(\theta_k)$

Key idea: natural gradient descent (NGD)

Gradient descent: $\theta_{k+1} = \theta_k - \alpha_k \nabla f(\theta_k)$

Newton method: $\theta_{k+1} = \theta_k - \alpha_k \left[\nabla^2 f(\theta_k)\right]^{-1} \nabla f(\theta_k)$

Natural gradient: $\theta_{k+1} = \theta_k - \alpha_k \left[F(\theta_k)\right]^{-1} \nabla f(\theta_k)$

if $f(\theta) = -\log p(x;\theta)$

$$F(\theta_k) = \mathbb{E}\left[\nabla_\theta \log p(x;\theta) \nabla_\theta \log p(x;\theta)^T\right]$$

Key idea: natural gradient descent (NGD)

Gradient descent: $\theta_{k+1} = \theta_k - \alpha_k \nabla f(\theta_k)$

Newton method: $\theta_{k+1} = \theta_k - \alpha_k \left[\nabla^2 f(\theta_k)\right]^{-1} \nabla f(\theta_k)$

Natural gradient: $\theta_{k+1} = \theta_k - \alpha_k \left[F(\theta_k)\right]^{-1} \nabla f(\theta_k)$

if $f(\theta) = -\log p(x;\theta)$

$$F(\theta_k) = \mathbb{E}\left[\nabla_\theta \log p(x;\theta) \nabla_\theta \log p(x;\theta)^T\right]$$

like Newton method for optimization of probability densities

How to approximate NGD?

Natural gradient: $\theta_{k+1} = \theta_k - \alpha_k \left[F(\theta_k)\right]^{-1} \nabla f(\theta_k)$

How to approximate NGD?

Natural gradient: $\theta_{k+1} = \theta_k - \alpha_k \left[F(\theta_k)\right]^{-1} \nabla f(\theta_k)$

Explicitly approximating and inverting $F(\theta_k)$

How to approximate NGD?

Natural gradient: $\theta_{k+1} = \theta_k - \alpha_k \left[F(\theta_k)\right]^{-1} \nabla f(\theta_k)$

Explicitly approximating and inverting $F(\theta_k)$

- Cholesky factorization [Groose et al, 2015]

How to approximate NGD?

Natural gradient: $\theta_{k+1} = \theta_k - \alpha_k \left[F(\theta_k)\right]^{-1} \nabla f(\theta_k)$

Explicitly approximating and inverting $F(\theta_k)$

- Cholesky factorization [Groose et al, 2015]
- Kronecker factored approximation [Martens et al, 2015]
- …

How to approximate NGD?

Natural gradient: $\theta_{k+1} = \theta_k - \alpha_k \left[F(\theta_k)\right]^{-1} \nabla f(\theta_k)$

Explicitly approximating and inverting $F(\theta_k)$

Change features or parameters so that $F(\theta_k) \approx I$

- Cholesky factorization [Groose et al, 2015]
- Kronecker factored approximation [Martens et al, 2015]
- …

How to approximate NGD?

Natural gradient: $\theta_{k+1} = \theta_k - \alpha_k \left[F(\theta_k)\right]^{-1} \nabla f(\theta_k)$

Explicitly approximating and inverting $F(\theta_k)$

- Cholesky factorization [Groose et al, 2015]
- Kronecker factored approximation [Martens et al, 2015]
- …

Change features or parameters so that $F(\theta_k) \approx I$

- Normalization of input [LeCun et al, 1991]

How to approximate NGD?

Natural gradient: $\theta_{k+1} = \theta_k - \alpha_k \left[F(\theta_k)\right]^{-1} \nabla f(\theta_k)$

Explicitly approximating and inverting $F(\theta_k)$

- Cholesky factorization [Groose et al, 2015]
- Kronecker factored approximation [Martens et al, 2015]
- …

Change features or parameters so that $F(\theta_k) \approx I$

- Normalization of input [LeCun et al, 1991]
- Batch normalization [Ioffe et al, 2015]

How to approximate NGD?

Natural gradient: $\theta_{k+1} = \theta_k - \alpha_k \left[F(\theta_k)\right]^{-1} \nabla f(\theta_k)$

Explicitly approximating and inverting $F(\theta_k)$

- Cholesky factorization [Groose et al, 2015]
- Kronecker factored approximation [Martens et al, 2015]
- …

Change features or parameters so that $F(\theta_k) \approx I$

- Normalization of input [LeCun et al, 1991]
- Batch normalization [Ioffe et al, 2015]
- Weight normalization [Salimans et al, 2015]

How to approximate NGD?

Natural gradient: $\theta_{k+1} = \theta_k - \alpha_k \left[F(\theta_k)\right]^{-1} \nabla f(\theta_k)$

Explicitly approximating and inverting $F(\theta_k)$

- Cholesky factorization [Groose et al, 2015]
- Kronecker factored approximation [Martens et al, 2015]
- …

Change features or parameters so that $F(\theta_k) \approx I$

- Normalization of input [LeCun et al, 1991]
- Batch normalization [Ioffe et al, 2015]
- Weight normalization [Salimans et al, 2015]
- Normalization propagation [Arpit et al, 2016]
- …

How to approximate NGD?

Natural gradient: $\theta_{k+1} = \theta_k - \alpha_k \left[F(\theta_k)\right]^{-1} \nabla f(\theta_k)$

Explicitly approximating and inverting $F(\theta_k)$

- Cholesky factorization [Groose et al, 2015]
- Kronecker factored approximation [Martens et al, 2015]
- …

Change features or parameters so that $F(\theta_k) \approx I$

- Normalization of input [LeCun et al, 1991]
- Batch normalization [Ioffe et al, 2015]
- Weight normalization [Salimans et al, 2015]
- Normalization propagation [Arpit et al, 2016]
- …

Other connections

- AdaGrad as approximation to NGD [Wager et al, 2013]

How to approximate NGD?

Natural gradient: $\theta_{k+1} = \theta_k - \alpha_k \left[F(\theta_k)\right]^{-1} \nabla f(\theta_k)$

Explicitly approximating and inverting $F(\theta_k)$

- Cholesky factorization [Groose et al, 2015]
- Kronecker factored approximation [Martens et al, 2015]
- …

Change features or parameters so that $F(\theta_k) \approx I$

- Normalization of input [LeCun et al, 1991]
- Batch normalization [Ioffe et al, 2015]
- Weight normalization [Salimans et al, 2015]
- Normalization propagation [Arpit et al, 2016]
- …

Other connections

- AdaGrad as approximation to NGD [Wager et al, 2013]
- Dropout as L_2 regularization in the Fisher space [Wager et al, 2013]
- …

How to approximate NGD?

Natural gradient: $\theta_{k+1} = \theta_k - \alpha_k \left[F(\theta_k)\right]^{-1} \nabla f(\theta_k)$

Explicitly approximating and inverting $F(\theta_k)$

- Cholesky factorization [Groose et al, 2015]
- Kronecker factored approximation [Martens et al, 2015]
- …

Change features or parameters so that $F(\theta_k) \approx I$

- Normalization of input [LeCun et al, 1991]
- Batch normalization [Ioffe et al, 2015]
- Weight normalization [Salimans et al, 2015]
- Normalization propagation [Arpit et al, 2016]
- …

Other connections

- AdaGrad as approximation to NGD [Wager et al, 2013]
- Dropout as L_2 regularization in the Fisher space [Wager et al, 2013]
- …

Variable Selection in High Dimensional Feature Space

A combination of Sure Independence Screening (SIS) and Smoothly Clipped Absolute Deviation (SCAD) method

Biwei Huang[1] Xiongtao Ruan[2]

[1]Department of Philosophy

[2]Department of Computational Biology
School of Computer Science

May 4th, 2017

Problem setting and SIS

- Problem: $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, where $\mathbf{y}, \boldsymbol{\epsilon} \in \mathbb{R}^n$, $\mathbf{X} \in \mathbb{R}^{n \times p}$, $\boldsymbol{\beta} \in \mathbb{R}^p$
- Aim: select $o(n)$ variables from ultra-high dimensional feature space ($p = O(e^{n^\xi})$, for $\xi > 0$).
- Two steps: 1. use SIS to reduce the dimensions to $d = \Theta(n)$; 2. use SCAD method to further reduce the dimensions.
- SIS is based on correlation learning: $\boldsymbol{\omega} = \boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y}$.
- $\mathcal{M}_\gamma = \{1 \leq i \leq p : |\omega_i| \text{ is among the first } [\gamma n] \text{ largest of all}\}$, where $\gamma \in (0, 1)$, $[\gamma n]$ means the integer part of γn.
- Main property: Under regularity condition with sparsity assumption, SIS is accurate with high probability:

$$P(\mathcal{M}_* \subset \mathcal{M}_\gamma) = 1 - O[\exp\{-Cn^{1-2\kappa/\log(n)}\}]$$

where $\mathcal{M}_*$ is the true model, $C > 0$, and $\kappa \in (0, \frac{1}{2})$ is a parameter.

SCAD

- Problem: $\| \mathbf{y} - \mathbf{X}\boldsymbol{\beta} \|_2^2 / 2 + \sum_{j=1}^{d} p_{\lambda_j}(\beta_j)$.
- The SCAD penalty is defined as

$$p_\lambda(|\beta|) = \begin{cases} \lambda|\beta| & \text{if } 0 \le |\beta| \le \lambda; \\ -\frac{|\beta|^2 - 2a\lambda|\beta| + \lambda^2}{2(a-1)} & \text{if } \lambda \le |\beta| \le a\lambda; \\ (a+1)\lambda^2/2 & \text{if } |\beta| \ge a\lambda, \end{cases}$$

  for some $a > 2$.
- With SCAD penalty, the estimator has three properties: sparsity, unbiasedness, and continuity.
- The estimator is root-n consistent ($O_p(n^{-1/2})$) if $\lambda_j \to 0$ for all j.
- The estimator has oracle properties. And SIS-SCAD also has oracle properties

Reference

Fan, Jianqing and Lv, Jinchi, "Sure independence screening for ultrahigh dimensional feature space", *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2008; 70(5), pp. 849–911.

Fan, Jianqing and Li, Runze, "Variable selection via nonconcave penalized likelihood and its oracle properties", *Journal of the American statistical Association*, 2001; 96(456), pp. 1348–1360.

Fan, Jianqing and Song, Rui and others, "Sure independence screening in generalized linear models with NP-dimensionality", *The Annals of Statistics*, 2010; 38(6), pp. 3567–3604.

Tuning parameter selection in high dimensional variables selection tasks

Yu Chen, Haohan Wang

Challenges	Solutions
some tuning parameter has unknown components $\lambda \sim \frac{\sigma \|\|X^T \epsilon\|\|_\infty}{\sqrt{n}}$	Tuning insensitive or tuning free approaches: TIGER, Square-Root Lass, B-TREX
data dimension ***d*** is scaled with data sample size ***n***	Extended information criteria

CONSISTENCY AND CONVERGENCE RATES OF GAUSSIAN RBF KERNEL SVMS

XIAOQI CHAI, JIAMING CAO

GAUSSIAN RBF KERNEL SVMS

- Training Data $T = ((x_i, y_i)), x_i \in \mathbb{R}^d, y_i \in \{-1, 1\}, i = 1, \ldots, n, i.i.d.$
- Gaussian RBF Kernel $k_\sigma(x, x') := \frac{1}{(\sqrt{2\pi}\sigma)^d} \exp(\frac{-||x - x'||^2}{2\sigma^2}), \sigma > 0$
- SVM as an optimization problem

$$f_{T,\lambda} = \arg \min_{f \in \mathcal{H}_k} \lambda ||f||_k^2 + \frac{1}{n} \sum_{i=1}^{n} (1 - y_i f(x_i))_+$$

CONSISTENCY

Definition 1 *A classifier f is consistent if $R_P(f_T)$ converges to R_P^* in probability, i.e. for every $\epsilon > 0$,*

$$\lim_{n \to \infty} Pr(R_P(f_T) - R_P^* > \epsilon) = 0$$

Definition 2 *A classifier is universally consistent if it is consistent for any distribution P on (X, Y).*

CONSISTENCY

Theorem 1 *Let $X \subset \mathbb{R}^d$ be compact and k be a universal kernel on X with $\mathcal{N}((X, d_k), \varepsilon) \in O(\varepsilon^{-\alpha})$ for some $\alpha > 0$. Suppose that we have a positive sequence (λ_n) with $\lambda_n \to 0$ and $\lambda_n \in O(n^{-\beta})$ for some $0 < \beta < \frac{1}{\alpha}$. Then for all Borel probability measures P on $X \times Y$ and all $\varepsilon > 0$ we have*

$$\lim_{n \to \infty} \Pr(\{T \in (X \times Y)^n : R_P(f_{T,\lambda_n}) \leq R_P + \varepsilon\}) = 1$$

Comments

- Gaussian RBF kernel is a universal kernel — linear combinations of this kernel can approximate any continuous function on the input space.
- When using Gaussian RBF kernel, we can choose $\alpha = d$.
- Gaussian RBF kernel SVMs are universally consistent with appropriate choice of the parameter λ.
- But there does not exist a universal convergence rate for all probability.

PROBABILITY ASSUMPTIONS

Tsybakov noise *Let $0 \leq q \leq \infty$ and P be a probability measure on (X, Y). We say that P has Tsybokov noise exponent q if there exists a constant $C > 0$ such that for all sufficiently small $t > 0$ we have*

$$P_X(|2\eta(x) - 1| \leq t) \leq Ct^q$$

where $\eta(x) := P(y = 1|x)$.

Geometric noise *Let X is a compact subset of $\mathbb{R}^d$, and P be a probability measure on (X, Y). We say that P has geometric noise exponent α if there exists a constant $C > 0$ such that for all sufficiently small $t > 0$ we have*

$$\int_x |2\eta(x) - 1| e^{-\frac{\tau(x)^2}{t}} P_X(dx) \leq Ct^{\frac{\alpha d}{2}}$$

where $\tau(x)$ is a function that measures the distance of x to the decision boundary.

<u>*Comments*</u>

- They both describe noise near the decision boundary.
- They can be related by the envelope order of the probability.

CONVERGENCE RATE 1

Theorem 2 *Let X be the closed unit ball of $\mathbb{R}^d$, and P be a distribution on $X \times Y$ with Tsybakov noise exponent $q \in [0, \infty]$ and geometric noise exponent $\alpha \in (0, \infty)$. Define,*

$$\beta := \begin{cases} \frac{\alpha}{2\alpha+1}, & if \alpha \leq \frac{q+2}{2q} \\ \frac{2\alpha(q+1)}{2\alpha(q+2)+3q+4} & othersise \end{cases}$$

and $\lambda_n := n^{-(\alpha+1)/\alpha\beta}$, $\sigma_n := n^{\beta/\alpha d}$. Then,
$\forall \epsilon > 0$, $\exists C > 0$ such that $\forall x \geq 1$ and $n \geq 1$ SVM_0 with Gaussian RBF kernel k_{σ_n} satisfies,

$$\Pr(T \in (X \times Y)^n : R_P(f_{T,\lambda_n}) \leq R_P + Cx^2 n^{-\beta+\epsilon}) \geq 1 - e^{-x}$$

If $\alpha = \infty$ the inequality holds if $\sigma_n \equiv \sigma > 2\sqrt{d}$.

Comment

- Under conditions of $\alpha \to \infty$ and $q \to \infty$, optimal rate can be achieved, which approaches $O(\frac{1}{n})$.

CONVERGENCE RATE 2

Theorem 3 *Let H be a RKHS of a continuous kernel on a compact metric space X with complexity exponent $0 < p < 2$, and let P be a probability measure on (X, Y) with Tsybakov noise exponent $0 \le q \le \infty$. Furthermore, assume that H approximates P with exponent $0 < \beta \le 1$. We define $\lambda := n^{-\alpha}$ for some $\alpha \in (0,1)$ and all $n \ge 1$. If $\alpha < \frac{4(q+1)}{(2q+pq+4)(1+\beta)}$ then there exists a $C > 0$ with*

$$Pr(R_P(f_{T,\lambda_n}) \le R_P + Cx^2 n^{-\alpha\beta}) \ge 1 - e^{-x}$$

for all $n \ge 1$ and $x \ge 1$. Here Pr is the outer probability of P^n in order to avoid memorability considerations. Furthermore, if $\alpha \ge \frac{4(q+1)}{(2q+pq+4)(1+\beta)}$ then for all $\epsilon > 0$ there exists a $C > 0$ such that for all $x \ge 1, n \ge 1$ we have

$$Pr(R_P(f_{T,\lambda_n}) \le R_P + Cx^2 n^{-\frac{4(q+1)}{2q+pq+4}+\alpha+\epsilon}) \ge 1 - e^{-x}$$

Comments

- This rate is derived under more general condition
- The best rate it can achieve is $O(n^{-\frac{4\beta(q+1)}{(2q+pq+4)(1+\beta)}})$

DISCUSSION

- Gaussian RBF kernel SVMs are universally consistent;
- and have optimal fast convergence rate $O(\frac{1}{n})$ under some assumptions,
- which restrict the noisiness of the data-generating distribution near the decision boundary.
- More general convergences rates in report

Learning with Conditional Random Fields (CRF)

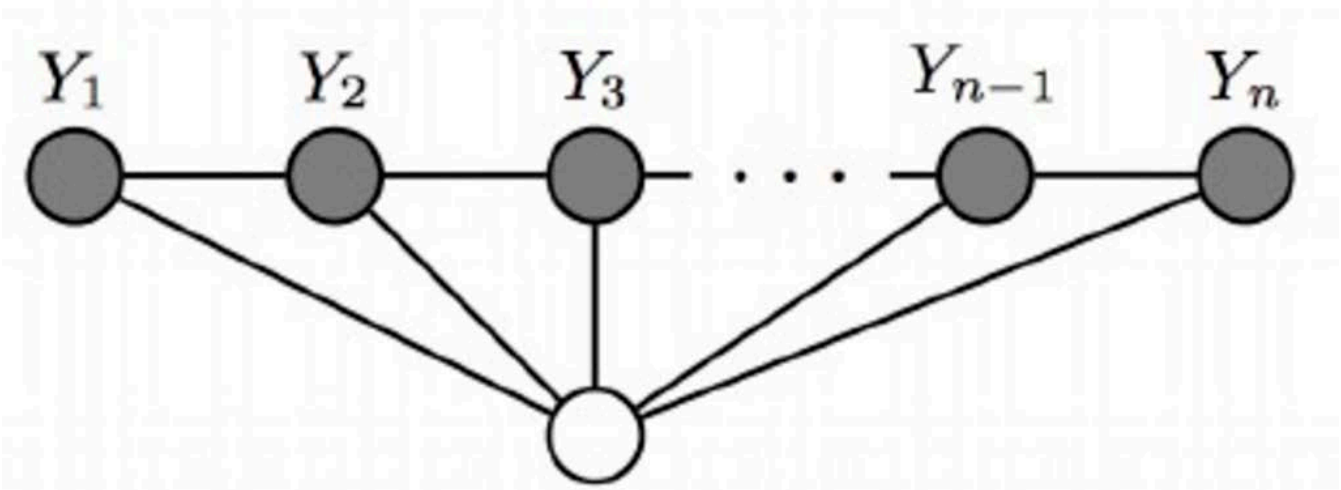
Chia-Yin Shih

Liang-Kang Huang

Brief introduction of CRF

- **Brief intro:** a discriminative learning framework encoded with undirected PGM
- **Motivation:** Allows for more realistic modeling of NLP and image recognition tasks
- **702 Topics:** Probabilistic graphical model, Reproducing Kernel Hilbert Space, MLE/MAP

Formulation of CRF



The conditional probability has the following form:

$$P_\theta(Y|x) = \frac{1}{Z_x(\theta)} exp\{\sum_{i=1}^{k} \theta_i f_i(x, Y)\}$$

We estimate the parameters θ.

Learning with CRF

- Given observations (x_1, y_1), (x_2, y_2) … (x_n, y_n), how to estimate θ, such that given unseen x_k, we can predict y_k by evaluating:

$$P_\theta(Y|x) = \frac{1}{Z_x(\theta)} exp\{\sum_{i=1}^{k} \theta_i f_i(x, Y)\}$$

MLE and MAP of CRF

- MLE

$$l_{Y|X}(\theta : D) = \sum_{m=1}^{M} \ln \frac{1}{Z_x(\theta)} \tilde{P}_\theta(x[m], y[m])$$

$$\begin{aligned}\frac{\partial}{\partial \theta_i} l_{Y|X}(\theta : D) &= \frac{\partial}{\partial \theta_i} \sum_{m=1}^{M} \{\sum_i \theta_i f_i(x[m], y[m]) - \ln Z_{x[m]}(\theta)\} \\ &= \sum_{m=1}^{M} f_i(x[m], y[m]) - E_\theta[f_i(x[m], Y)]\end{aligned}$$

- MAP

$$\begin{aligned}\arg\max_\theta P(D, \theta) &= \arg\max_\theta P(D|\theta)P(\theta) \\ &= \arg\max_\theta (l(\theta : D) + logP(\theta))\end{aligned}$$

Gaussian: $P(\theta : \sigma^2) = \prod_{i=1}^{k} \frac{1}{\sqrt{2\pi}\sigma} \exp\{-\frac{\theta_i^2}{2\sigma^2}\}$

Laplacian: $P(\theta : \beta) = \prod_{i=1}^{k} \frac{1}{2\beta} \exp\{-\frac{|\theta_i|}{\beta}\}$

The above formulation of ML/MAP learning can be kernelized using the Representer Theorem.

Kernel CRF

- Representer Theorem for CRF

Objective Function: $$R_\phi f = \sum_{i=1}^{n} \phi\left(\boldsymbol{y}^{(i)}, f(\mathfrak{g}^{(i)}, \boldsymbol{x}^{(i)})\right) + \Omega\left(\|f\|_K\right)$$

Mercer Kernel: $$K((\mathfrak{g}, \boldsymbol{x}, c, \boldsymbol{y}_c), (\mathfrak{g}', \boldsymbol{x}', c', \boldsymbol{y}'_{c'})) \in \mathbb{R}$$

The solution has the following form:

$$f^\star(\cdot) = \sum_{i=1}^{n} \sum_{c \in \mathcal{C}(\mathfrak{g}^{(i)})} \sum_{\boldsymbol{y}_c \in \mathcal{Y}^{|c|}} \alpha_c^{(i)}(\boldsymbol{y}_c)\, K_c(\boldsymbol{x}^{(i)}, \boldsymbol{y}_c; \cdot)$$

Study on Structured Sparsity

Yang Yang

May 4, 2017

Sparse variable selection has wide applications

-- signal processing, computer vision, computational biology…

Standard sparsity: consider individual variables equally

Structured sparsity: utilize spatial structures of variables

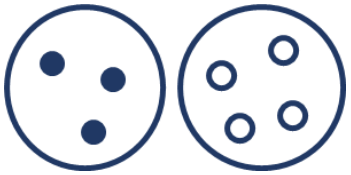
$$y = X\bar{\beta} + \epsilon \qquad \hat{\beta} = \arg\min_{\beta} \frac{1}{N}\sum_{i=1}^{N} L(\beta, x_i, y_i) + \lambda\Omega(\beta)$$

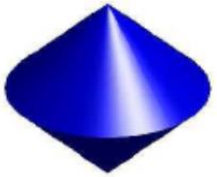
Standard Lasso

$$\Omega_{L_1}(\beta) = \| \beta\|_1$$

Group Lasso with disjoint groups

$$\Omega^{\mathcal{G}}_{\text{group}}(\beta) = \sum_{k=1}^{m} \lambda_k \| \beta_{G_k} \|_2 \qquad \mathcal{G} = \{G_1, \cdots, G_M\}$$





l_1/l_2 norm ball without overlap

l_1/l_2 norm ball with overlap

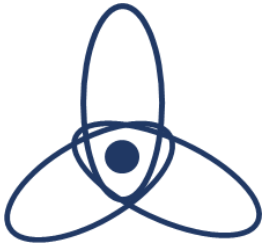
Bach et al. 2012

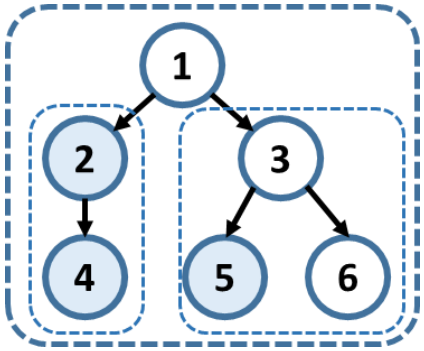
Overlapping group lasso

$$\Omega^{\mathcal{G}}_{\text{overlap}}(\beta) = \inf_{v \in \mathcal{V}_{\mathcal{G}}} \sum_{G \in \mathcal{G}} \lambda_G \| v_G \|, \ \ s.t. \sum_{G \in \mathcal{G}} v_G = \beta, v_G \in \mathbb{R}^p$$

Hierarchical sparsity

$$\Omega_{\mathcal{G},\gamma}(\beta) = \sum_{k=1}^{m} | N_k |^{\gamma_0}, \ N_k = \| \beta_{\mathcal{G}_k} \|_{\gamma_k}$$





Optimization methods

Proximal methods

Network flow algorithms

Block-coordinate Descent Algorithms

Group Lasso with disjoint groups

Nardi (2011): group Lasso is asymptotically optimal; offers the same asymptotic guarantees as the least squares estimator

- model selection consistency
- estimation consistency

Theorem 1: Let $\lambda_G = 1/\|\beta_h^{OLS}\|_2^{\gamma}$, for some $\gamma > 0$, such that $n^{(\gamma+1)/2}\lambda \to \infty$, if $\sqrt{n}\lambda \to 0$

$\mathbb{P}\{\mathcal{H} = \mathcal{H}_0\} \to 1, \text{ as } n \to \infty$ $\quad \mathcal{H} = \{h : \hat{\beta}_h \neq 0\}$

$\sqrt{n}(\beta - \bar{\beta}) \xrightarrow{d} Z$ $\quad Z_{\mathcal{G}_0} \sim \mathcal{N}(0, \sigma^2 M_{\mathcal{G}_0}^{-1}), \text{ and } Z_{\mathcal{G}_0^c} = 0$

Huang et al.(2011) : under strongly group-sparse assumption, the group Lasso has superior performance over the standard Lasso

(g,k) strongly group-sparse: $\text{supp}(\bar{\beta}) \subset G_S, |G_S| \leq k, |S| \leq g$

Theorem 2: $\|\beta - \bar{\beta}\|_2 \leq \frac{c_0}{\rho_-(s)\sqrt{n}}(1 + 0.25c^{-1})\sqrt{A^2k + gB^2}$

Theorem 3: $\sup_{\bar{\beta} \in H(g,k)} \mathbb{E}_y \|\beta - \bar{\beta}\|_2^2 \geq \sigma^2 \frac{\rho_-(2g)}{24n\rho_+(2g)^2}[g \ln((m-g)/g) - (g+2)\ln 4]$

		Standard Lasso
Upper bound:	$\|\hat{\beta} - \bar{\beta}\|_2^2 = O((k + g\ln(m/\eta)/n)$	$\|\hat{\beta}_{L_1} - \bar{\beta}\|_2^2 = O((\|\bar{\beta}\|_0 \ln(p/\eta))/n)$
Lower bound:	$\|\hat{\beta} - \bar{\beta}\|_2^2 = \Omega((k + g\ln(m/g))/n)$	$\|\hat{\beta}_{L_1} - \bar{\beta}\|_2^2 = \Omega(k\ln(p/k)/n)$

Overlapping Group Lasso

Obozinski et al. (2011): support recovery - the support of $\hat{\beta}$ matches the support of $\bar{\beta}$ with high probability

Theorem 4: $\mathcal{G}_1^*(\hat{\beta}) \subset \mathcal{G}_1(\bar{\beta})$ with probability approaching to 1

Theorem 5: $\mathcal{G}_1^*(\bar{\beta}) \subset \mathcal{G}_1^*(\hat{\beta}) \subset \mathcal{G}_1(\bar{\beta})$ with high probability

Percival et al.(2011): overlapping group Lasso shares many of the same theoretical guarantees as the group Lasso, if the sets of groups are not too complex.

Theorem 6:
$$\frac{1}{n}\| X(\hat{\beta} - \bar{\beta})\| \leq \frac{64\sigma^2}{\kappa^2\sqrt{n}}\left(\max_{G\in\mathcal{G}} |G| + A\sqrt{\max_{G\in\mathcal{G}} |G|}\log M\right)$$
$$\| \hat{\beta} - \bar{\beta}\|_{2,1,\mathcal{G}} \leq \frac{32\sigma}{\kappa\sqrt{n}}\left(\max_{G\in\mathcal{G}} |G| + A\sqrt{\max_{G\in\mathcal{G}} |G|}\log M\right)^{1/2}$$

Hierarchical sparsity

Zhao et al.(2009): Composite Absolute Penalties (CAP) family; particular overlapping patterns of groups are designed; with-in group and group-wise sparsity imposed

Thank you!

Word embeddings, how do they work?

Ezra Winston
StatML, spring 2017
CMU

May 4, 2017

Word Embeddings

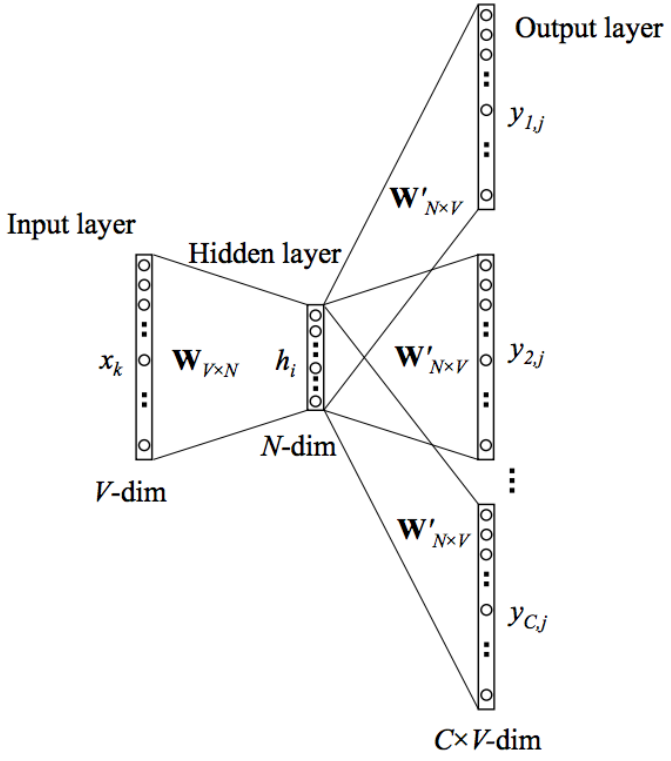
- Given vocabulary $w_1, w_2, \ldots, w_n$, word embeddings are vectors $v_1, v_2, \ldots, v_n \in \mathbb{R}^d$.
- Goal is to capture the semantics of words in a low dimensional space. Typically $d \approx 300$ while $n \approx 10^5$.
- Shown to improve performance when used as features in NLP tasks, compared to, say, one-hot encoding of words where $d = n$.

Example: word2vec (Mikolov et al. 2013)

- Embeddings v_i learned on a large text corpus by modeling probability of word w_i appearing given context word w_j.
- Approximates softmax objective

$$P(w_j|w_i) = \frac{\exp(v_j^\top h_i)}{\sum_{k=1}^{n} \exp(v_k^\top h_i)}$$

where h_i are *context* embeddings and v_i are *output* embeddings.



Theoretical Questions

- Under what assumptions are popular embedding methods consistent?

Theoretical Questions

- Under what assumptions are popular embedding methods consistent?
- Why do low dimensional vectors capture the semantics well?

Theoretical Questions

- Under what assumptions are popular embedding methods consistent?
- Why do low dimensional vectors capture the semantics well?
- Many more: relations $\approx$ directions, syntax words, polysemy...

Explanation 1: Arora et al. 2016

Setup:

- Language generation is a random walk of a unit-length discourse vector $c_t \in \mathbb{R}^d$
- Embedding vectors are approximately uniformly distributed in space.
- c_t is a slow random walk: stationary distribution has $\|c_t - c_{t-1}\|_2 < \epsilon_2/\sqrt{d}$.
- The probability of word w being emitted at time t is $p(w|c_t) \propto \exp(\langle c_t, v_w \rangle)$.

Explanation 1: Arora et al. 2016

Setup:

- Language generation is a random walk of a unit-length discourse vector $c_t \in \mathbb{R}^d$
- Embedding vectors are approximately uniformly distributed in space.
- c_t is a slow random walk: stationary distribution has $\|c_t - c_{t-1}\|_2 < \epsilon_2/\sqrt{d}$.
- The probability of word w being emitted at time t is $p(w|c_t) \propto \exp(\langle c_t, v_w \rangle)$.

Theorem

$$\log p(w_i, w_j) = \frac{\|v_i + v_j\|}{2d} - 2\log Z \pm \epsilon$$

where Z is a constant that is close to the partition function and $\epsilon = \tilde{O}(\frac{1}{\sqrt{n}}) + \tilde{O}(\frac{1}{d}) + O(\epsilon_2)$.

Explanation 2: Hashimoto et al. 2016

Given a random walk over embedding vectors that tends to move to nearby vectors, we can recover the vector distances from the statistics of the walk.

Setup:

- $P(w|w') = h(\frac{1}{\sigma}\|v_w - v_{w'}\|_2^2)$ for subgaussian function h

Explanation 2: Hashimoto et al. 2016

Given a random walk over embedding vectors that tends to move to nearby vectors, we can recover the vector distances from the statistics of the walk.

Setup:

- $P(w|w') = h(\frac{1}{\sigma}\|v_w - v_{w'}\|_2^2)$ for subgaussian function h

Theorem

Let $C_{ij}^{m,n}(t_n)$ be the number of times word w_j occurs t_n words after w_i. There exists a_i and b_j such that simultaneously over all i, j:

$$\lim_{n\to\infty} -\log(C_{ij}^{m,n}(t_n)) - a_i^{m,n} - b_j^{m,n} \to \|x_i - x_j\|_2^2$$

Graph-Based Semi-Supervised Classification and the Graph Construction

Xuan Wu
Institute for Software Research
Carnegie Mellon University

Motivation

- Labeled data is very few and expensive to obtain
- Unlabeled data is widely available, but barely utilized
- Graph-based SSL methods provide uniform representation for heterogeneous data, and are parallelizable, scalable to large data
- In many tasks, we need to first construct the similarity graph
- Different graphs can lead to different classification accuracies

Gaussian Random Fields Method

- GRF defined as follows:

$$P_\beta(f) = \frac{e^{-\beta E(f)}}{Z_\beta}$$, where $$E(f) = \frac{1}{2}\sum_{i,j} w_{ij}(f(i) - f(j))^2$$

To maximize $P_\beta(f)$ subject to $f(i) = f_l(i) = y_i$ for all labeled data, the solution is harmonic, satisfying

$\Delta f = 0$ on unlabeled data set, and $\Delta f = f_l$ on labeled data.

And f_u is given by $(I - P_{uu})^{-1} P_{ul} f_l$

Global and Local Consistency Method

1. Form the affinity matrix W defined by $W_{ij} = \exp(-\|x_i - x_j\|^2/2\sigma^2)$ if $i \neq j$ and $W_{ii} = 0$.
2. Construct the matrix $S = D^{-1/2}WD^{-1/2}$ in which D is a diagonal matrix with its (i,i)-element equal to the sum of the i-th row of W.
3. Iterate $F(t+1) = \alpha SF(t) + (1-\alpha)Y$ until convergence, where α is a parameter in $(0,1)$.
4. Let F^* denote the limit of the sequence $\{F(t)\}$. Label each point x_i as a label $y_i = \arg\max_{j \leq c} F^*_{ij}$.

Closed Form Solution: $F^* = (I - \alpha S)^{-1} Y$

K-NN / ε-Neighborhood Graph

- k-NN is more robust to scale and density while an inaccurate choice of ε in may result in disconnected graphs using ε-neighborhood
- In practice, k-NN usually performs better then ε neighborhood
- However, k-NN typically leads to different degrees on different nodes

Graph Construction with B-Matching

- The first step is graph sparsification, in which we select a subset of edges to satisfy

$$\min_{P \in \{0,1\}^{n \times n}} \sum_{ij} P_{ij} D_{ij}$$
$$\text{s.t} \sum_j P_{ij} = b,\ P_{ii} = 0,\ P_{ij} = P_{ji}, \forall i, j \in 1, ..., n$$

- The second step is Edge Re-Weighting. Available methods include Binary, Gaussian Kernel, and Locally Linear Reconstruction (LLR)

Graph Construction with Spectral Transform

- Consider the following form of kernel

$K = \sum_i \mu_i \phi_i \phi_i^T$, where $\phi_i s$ are the eigenvectors of the graph Laplacian L .

- We want to choose μ_i s.t the outer product of smaller eigenvalue has higher weight.
- To determine μ_i s, zhu et al. proposed to maximize the empirical kernel alignment

Thank You!

Sample Complexity of Reinforcement Learning Algorithms

Yifan Wu

yw4@andrew.cmu.edu

Reinforcement Learning

Markov Decision Processes with finite state & action space, finite time horizon $M = (\mathcal{S}, \mathcal{A}, R, P, H, \rho)$:

- $\mathcal{S}$: state space, $\mathcal{A}$: action space.
- R: reward function. H: time horizon in an episode.
- P: transition probability. ρ: initial state distribution.

An RL algorithm π sequentially maps the sample history to a policy μ^k, $k = 1, ..., T/H$, evaluated by its regret (stronger than risk):

$$\text{Regret}(T, \pi, M) = \sum_{k=1}^{T/H} V_M(\mu_M^*) - V_M(\mu^k) .$$

Bayes regret: assume M drawn from prior ϕ,

$$\text{BayesRegret}(T, \pi, \phi) = \mathbb{E}\left[\text{Regret}(T, \pi, M) | M \sim \phi\right] .$$

Algorithms (Model Based)

- Confidence set based algorithm:
  - Before each episode k, construct a confidence set $\mathcal{M}_k$ for M, act using the policy that is the most optimistic according to $\mathcal{M}_k$, update the confidence set.
  - Best known result: $\max_M \text{Regret}(T, \pi, M) = O(HS\sqrt{AT})$ with high prob.
  - A general proof technique:

$$\begin{aligned} V_M(\mu_M^*) - V_M(\mu^k) &= V_M(\mu_M^*) - V_{M_k}(\mu^k) + V_{M_k}(\mu^k) - V_M(\mu^k) \\ &\leq V_{M_k}(\mu^k) - V_M(\mu^k)\,. \end{aligned} \tag{1}$$

- Posterior sampling algorithm:
  - Before each episode k, sample an MDP M_k from the posterior distribution $\mathcal{H}_k$ for M, act using the optimal policy for M_k, update the posterior.
  - Best known result: $\text{BayesRegret}(T, \pi, \phi) = O(HS\sqrt{AT})$.
  - A general reduction: for any high prob. regret bound for a confidence set based algorithm **derived using** (1), the **same Bayes regret** holds for the posterior sampling algorithm.

Modeling Correlated Times Series Data

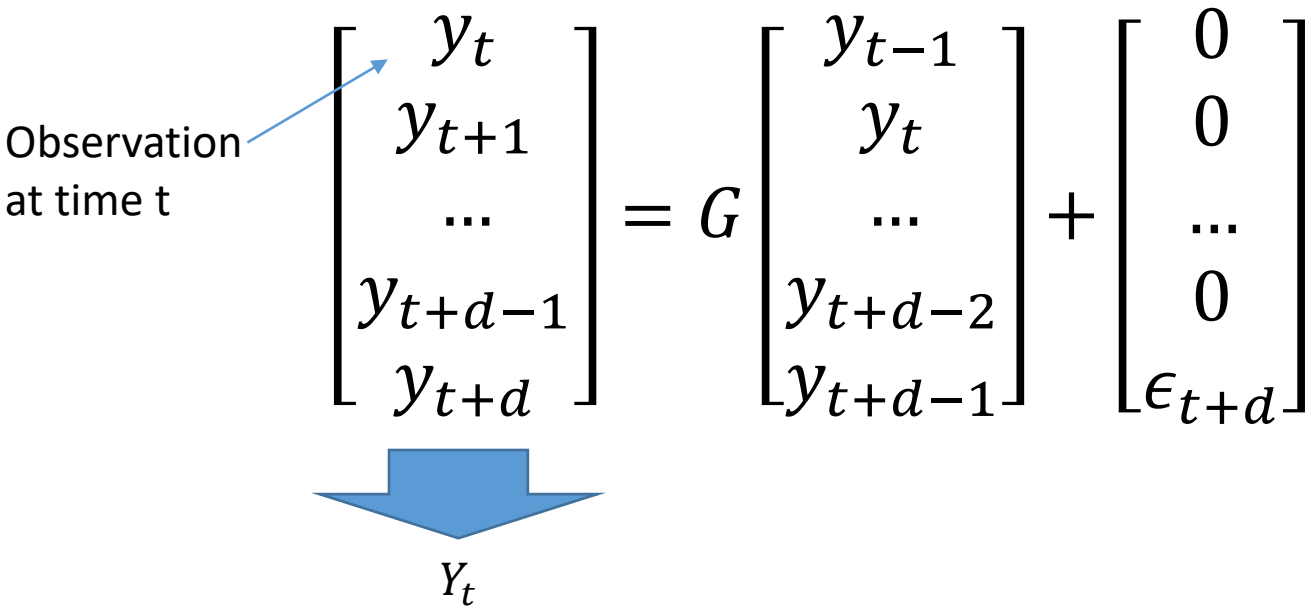
Chenghui Zhou

Predictive Linear Gaussian Model

(PLG)

$$\begin{bmatrix} y_t \\ y_{t+1} \\ \dots \\ y_{t+d-1} \\ y_{t+d} \end{bmatrix} = G \begin{bmatrix} y_{t-1} \\ y_t \\ \dots \\ y_{t+d-2} \\ y_{t+d-1} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dots \\ 0 \\ \epsilon_{t+d} \end{bmatrix}$$

Predictive Linear Gaussian Model



Predictive Linear Gaussian Model

$$\begin{bmatrix} y_t \\ y_{t+1} \\ \dots \\ y_{t+d-1} \\ y_{t+d} \end{bmatrix} = G \begin{bmatrix} y_{t-1} \\ y_t \\ \dots \\ y_{t+d-2} \\ y_{t+d-1} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dots \\ 0 \\ \epsilon_{t+d} \end{bmatrix}$$

Update matrix

Predictive Linear Gaussian Model

$$\begin{bmatrix} y_t \\ y_{t+1} \\ \dots \\ y_{t+d-1} \\ y_{t+d} \end{bmatrix} = G \begin{bmatrix} y_{t-1} \\ y_t \\ \dots \\ y_{t+d-2} \\ y_{t+d-1} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dots \\ 0 \\ \epsilon_{t+d} \end{bmatrix}$$

Update matrix

noise

Predictive Linear Gaussian Model

$$\begin{bmatrix} y_t \\ y_{t+1} \\ \dots \\ y_{t+d-1} \\ y_{t+d} \end{bmatrix} = G \begin{bmatrix} y_{t-1} \\ y_t \\ \dots \\ y_{t+d-2} \\ y_{t+d-1} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dots \\ 0 \\ \epsilon_{t+d} \end{bmatrix}$$

New observation

Update matrix

noise

Predictive Linear Gaussian Model

$$\begin{bmatrix} y_t \\ y_{t+1} \\ \dots \\ y_{t+d-1} \\ y_{t+d} \end{bmatrix} = G \begin{bmatrix} y_{t-1} \\ y_t \\ \dots \\ y_{t+d-2} \\ y_{t+d-1} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dots \\ 0 \\ \epsilon_{t+d} \end{bmatrix}$$

$$Cov\,(Y_{t-1}, \epsilon_{t+d}) = C$$

Relation with Other Models

Linear Dynamical System (LDS)

Transition matrix

$$x_t = T\, x_{t-1} + \sigma_t$$
$$y_t = O\, x_t + \delta_t$$

Observation matrix

$$\sigma_t \sim N(0, \Sigma)$$
$$\delta_t \sim N(0, \Delta)$$
$$x_0 \sim N(\mu_0, P_0)$$

Relation with Other Models

Linear Dynamical System (LDS)

$$x_t = T\ x_{t-1} + \sigma_t$$
$$y_t = O\ x_t + \delta_t$$

$$\sigma_t \sim N(0, \Sigma)$$
$$\delta_t \sim N(0, \Delta)$$
$$x_0 \sim N(\mu_0, P_0)$$

- Parameter G is a function of transition matrix and observation matrix
- Covariance C is a function of G, Δ, Σ and P_0
- Predictive Linear Gaussian Model is equivalent to LDS

Relation with Other Models

Autoregressive and Moving Average Model (ARMA)

$$y_t = \sum_{i=1}^{d} \Phi_i \, y_{t-i} + a_t + \sum_{i=1}^{p} \theta_i \, a_{t-i}$$

White noise

weights

Relation with Other Models

Autoregressive and Moving Average Model (ARMA)

$$y_t = \sum_{i=1}^{d} \Phi_i \, y_{t-i} + a_t + \sum_{i=1}^{p} \theta_i \, a_{t-i}$$

- Φ_i's can be seen as each entry in the update matrix G
- LDS subsumes ARMA model
- ARMA model can be represented by Predictive Linear Gaussian Model

Stochastic gradient method and its application

Yifan Sun

Deparment of Mathematical Sciences
Carnegie Mellon University

yifans@andrew.cmu.com

May 1, 2017

Overview

Vanilla SGD

In machine learning we often encounter the following problem:

$$\underset{x\in\mathbb{R}^d}{\text{minimize}} \left\{ F(x) = f(x) + \phi(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) + \phi(x) \right\}$$

where f(x) is usually the loss function and $\phi(x)$ is the regularization term. A lot of regularized empirical risk minimization problem could be written above form, for example Ridge regression, Lasso, SVM.

Vanilla SGD

For simplicity we assume there is no regularization term, and the usual gradient descent method is:

$$x^{(t)} \leftarrow x^{(t-1)} - \frac{\eta_t}{n} \sum_{i=1}^{n} \nabla f_i(x^{(t-1)})$$

Vanilla SGD

However, when n is large, the calculation of n derivatives in one iteration is very expensive. The vanilla stochastic gradient descent method choose one particular index uniformly from $i \in \{1, ...n\}$ and calculates:

$$x^{(t)} \leftarrow x^{(t-1)} - \eta_t \nabla f_i(x^{(t-1)})$$

More generally, the SGD updates has the form:

$$x^{(t)} \leftarrow x^{(t-1)} - \eta_t g(x^{(t-1)}) \tag{1}$$

$$\mathbb{E}[g(x^{(t-1)})] = \nabla f(x^{(t-1)}) \tag{2}$$

Vanilla SGD

The problem for SGD is its variance, because we randomly choose a $\nabla f_i(x)$ as our gradient and this quantity may have very large variance, it introduces too much noise and prohibits the algorithm from converging fast. The plain SGD converges at rate $\tilde{O}(\frac{L}{\epsilon})$ where L is the smoothness of f(x). It cannot converge at a faster rate even if $F(x)$ is strongly convex and smooth.

SVRG

Johnson and Zhang[2013] proposed a variance reduction method.The key idea is to take a snapshot vector $\tilde{x} = x_k$ as well as its full gradient $\tilde{\mu} = \nabla f(\tilde{x})$ once in every m iteration. The key observation is if we randomly pick one $i \in \{1...n\}$, we have $\mathbb{E}[\nabla f_i(\tilde{x}) - \tilde{\mu}] = 0$. Then we can add this term to our gradient update and get the update rule:

$$x^{(t)} \leftarrow x^{(t-1)} - \eta_t(\nabla f_i(x^{(t-1)}) - \nabla f_i(\tilde{x}) + \tilde{\mu})$$

SVRG

The magic here is that as $x^{(k)} \to x_*$, we have $\nabla f_i(x^{(k)}) \to \nabla f_i(\tilde{x})$ and $\tilde{\mu} = \nabla f(\tilde{x}) \to 0$ hence the variance of the update gradient $\tilde{\nabla}_k = \nabla f_i(x^{(k)}) - \nabla f_i(\tilde{x}) + \tilde{\mu}$ goes to 0. The iteration complexity is $O((n+\kappa)\log\frac{1}{\epsilon})$ where $\kappa = \frac{L}{\mu}$ is the condition number.

Acceleration of SGD

Nesterov's method could be used for deterministic gradient descent acceleration, the idea is to do update by:

$$y = x^{(k-1)} + \frac{k-2}{k+1}(x^{(k-1)} - x^{(k-2)})$$
$$x^{(k)} = y - t_k \nabla f(y)$$

where the $x^{(k-1)} - x^{(k-2)}$ term is a momentum term that makes us "go along the descending direction for more distance". In gradient descent this momentum accelerates the convergence, however in SGD since our updates is random, the update direction from last steps could deviate far from the full gradient and hence accumulates the incorrect momentum will lead to divergence of the algorithm.

Acceleration of SGD

Allen-Zhu[2017] proposed the state of art acceleration method for SGD called 'Katyusha' which converges at rate $O((n+\sqrt{n\kappa})\log\frac{1}{\epsilon})$ for L-smooth and α-strongly convex function. The update rule is:

$$x^{(k+1)} \leftarrow \tau_1 z_k + \tau_2 \tilde{x} + (1-\tau_1-\tau_2)y_k \tag{3}$$

$$\tilde{\nabla}_{k+1} \leftarrow \nabla f(\tilde{x}) + \nabla f_i(x_{k+1}) - \nabla f_i(\tilde{x}) \tag{4}$$

$$y^{(k+1)} \leftarrow x^{(k+1)} - \frac{1}{3L}\tilde{\nabla}_{k+1} \tag{5}$$

$$z^{(k+1)} \leftarrow z^{(k)} - \alpha\tilde{\nabla}_{k+1} \tag{6}$$

Acceleration of SGD

The key idea is to keep a snapshot of $\tilde{x}$, and by choosing $\tau_2 = 0.5$, in (3) the updates guarantees that the update of $x^{(k+1)}$ will not deviate too much from the snapshot. This can be seen as a 'negative momentum' which reduces the risks of going in the incorrect momentum direction too much. $z^{(k)}$ and $y^{(k)}$ in (3) give the 'momentum' term. (5) and (6) comes out of the idea of linearly coupling gradient descent and mirror descent algorithm in [Allen-Zhu and Orecchia 2014], which guarantees the algorithm to descend fast enough no matter $||\tilde{\nabla}_{k+1}||$ is large or small.

SGD on PCA

Shamir considers the problem of finding the largest eigenvector:

$$\min_{w \in \mathbb{R}^d : ||w||_2 = 1} -w^T \left(\frac{1}{n} \sum_{i=1}^{n} x_i x_i^T\right) w$$

SGD on PCA

The algorithm from [Shamir2015] keeps a snapshot $\tilde{w}$ and $\tilde{u} := \frac{1}{n}\sum_{i=1}^{n} x_i(x_i^T \tilde{w})$ for m periods, and within this m periods, it picks $i_t \in \{1, \ldots n\}$ uniformly, then the update rule is:

$$w_t' \leftarrow w_{t-1} + \eta_t(x_{i_t}x_{i_t}^T w_{t-1} - x_{i_t}x_{i_t}^T \tilde{w} + \tilde{u}) \tag{7}$$

$$w_t \leftarrow \frac{w_t'}{||w_t'||} \tag{8}$$

SGD on PCA

Theorem

With proper step choice, with high probability the stochastic gradient method in T steps gives a $\tilde{O}(\sqrt{p/T})$ optimal solution where p is a parameter depends on initialization of the algorithm:

- *If the algorithm is initialized from a warm start point w_0 satisfies $\frac{1}{\langle v, w_0 \rangle^2} \leq O(1)$ where v is the leading eigenvector we want to solve, then $p = O(1)$*
- *Under uniform random initialization $p = O(d)$.*
- *Using a more sophisticated random initialization $p = O(n_A)$ where n_A is the numerical rank of $A = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^T$ and $n_A \leq d$*

SGD on PCA

The proof of the theorem is very elaborate. It uses Azuma-Hoeffding inequality on martingales, conditional expectation technique and matrix algebra.

References

Allen-Zhu, Z. and Orrechia, L.

Linear coupling, an ultimate unification of gradient and mirror descent.

Informs Annual Meeting, 2014.

Allen-Zhu (2017)

Katyusha: The First Direct Acceleration of Stochastic Gradient Methods

In Symposium on Theory of Computing(STOC), 2017.

Johnson, R. and Zhang, T. (2013)

Accelerating stochastic gradient descent using predictive variance reduction.

In NIPS, 2013

In Symposium on Theory of Computing(STOC), 2017.

Shamir, O. (2015)

A Stochastic PCA and SVD Algorithm with an Exponential Convergence Rate

In ICML, 2015

Shamir, O. (2017)

Fast stochastic algorithms for SVD and PCA: convergence properties and convexity.

In ICML, 2017, doi:https://arxiv.org/pdf/1507.08788.pdf

Thanks!

A Survey of Theories on Generative Adversarial Networks

Chris Ying, Yijie Wang

May 4, 2017

GAN Minimax Value Function

$$\min_{\theta_g} \max_{\theta_d} V(D,G) = \mathbb{E}_{x \sim p_{data}}[\log D(x;\theta_d)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z;\theta_g);\theta_d))]$$

discriminator output on true data

discriminator output on generated data

$\boldsymbol{\theta}_g$= generator parameters

G = generator function

$\boldsymbol{\theta}_d$= discriminator parameters

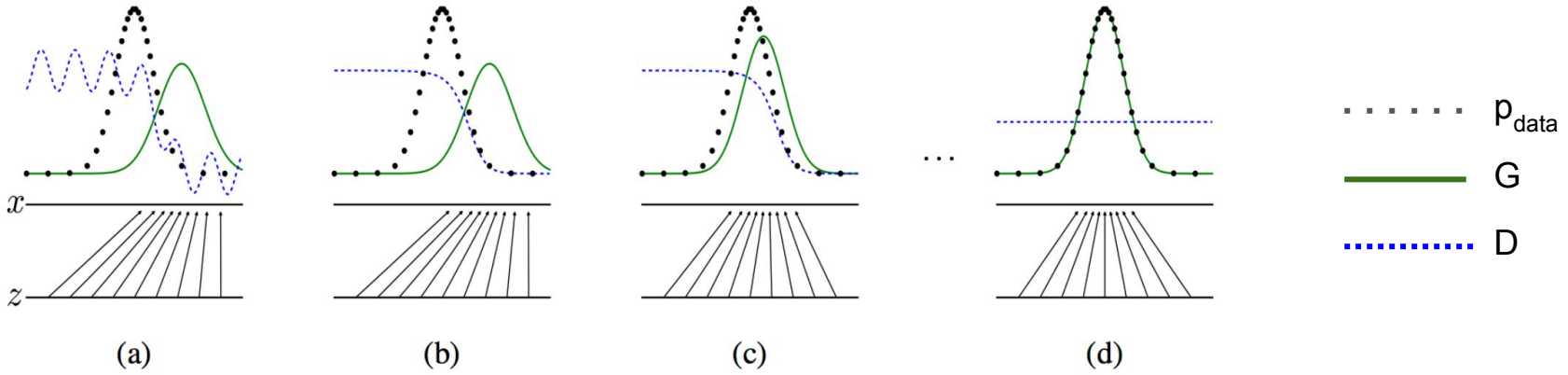
D = discriminator function

p_{data}= true distribution

p_z= noise distribution

Goodfellow, Ian, et al. "Generative adversarial nets." *Advances in neural information processing systems*. 2014.

Theoretical Results



Optimal D: $$D^*_G = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

At optimality: $$p_g = p_{data}$$

With *sufficient model capacity*, optimality is reached

Goodfellow, Ian, et al. "Generative adversarial nets." *Advances in neural information processing systems*. 2014.

Loss-Sensitive Generative Adversarial Networks on Lipschitz Densities

- Finite model capacity
- Lipschitz property assumption on the underlying distribution, on the model parameters and on the model inputs
- Adaptive margin:

$$L_\theta(x) \leq L_\theta(G_\phi(z)) - \Delta(x, G_\phi(z))$$

- Objective:

$$S(\theta, \phi^*) = \mathbb{E}_{x \sim P_{data}(x)} L_\theta(x) + \lambda \mathbb{E}_{x \sim P_{data}(x), z \sim P_z} (\Delta(x, G_{\phi^*}(z)) + L_\theta(x) - L_\theta(G_{\phi^*}(z)))_+$$

$$T(\theta^*, \phi) = \mathbb{E}_{z \sim P_z} L_{\theta^*}(G_\phi(z))$$

- Theoretical properties:
  - Convergence of generated distribution based on Nash Equilibrium
  - Probabilistically approximate generalization bound given for both the generator and the discriminator

Qi, Guo-Jun. "Loss-sensitive generative adversarial networks on lipschitz densities." *arXiv preprint arXiv:1701.06264* (2017).

Wasserstein GAN

- Uses Earth-Mover distance (a.k.a. Wasserstein-1 distance) instead of Jensen-Shannon divergence:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\big[\, \|x - y\| \,\big]$$

- New value function (assume generator fixed, *f* parametric family K-Lipschitz):

$$\max_{w \in \mathcal{W}} \mathbb{E}_{x\sim\mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z\sim p(z)}[f_w(g_\theta(z)]$$

- Desirable properties of Earth-Mover distance:
  - Continuous and differentiable almost everywhere
  - Guarantee of convergence similar to Jensen-Shannon divergence

Arjovsky, Martin, Soumith Chintala, and Léon Bottou. "Wasserstein gan." *arXiv preprint arXiv:1701.07875* (2017).

Predictive State Representation

An alternate approach to learn dynamical systems

Boyue Li

May 1, 2017

Language Technologies Institute

Introduction

1.1. Dynamical system

A dynamical system is a system whose

- output is decided by its state and noise
- state evolves with time over a state space
- state evolves according to a fixed rule

HMM and RNN are all dynamical systems[1, 2, 5].

More specifically,

$$s_{t+1} = f(s_t) \tag{1}$$

$$o_t = g(s_t) + \epsilon_t \tag{2}$$

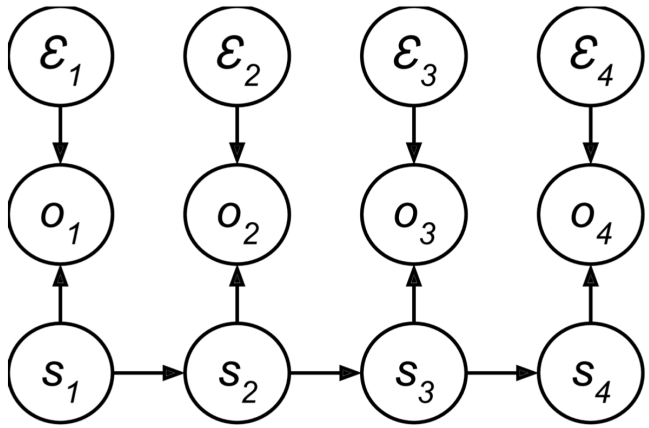


Figure 1: A dynamical system[3].

1.2. Spectral algorithm

Spectral algorithms factorize matrices of observable moments to learn a consistent model.

These algorithms are usually

- fast (comparing to E-M and RNN!)
- simple
- consistent
- globally optimum

Predictive state representation

2.1. Formulation

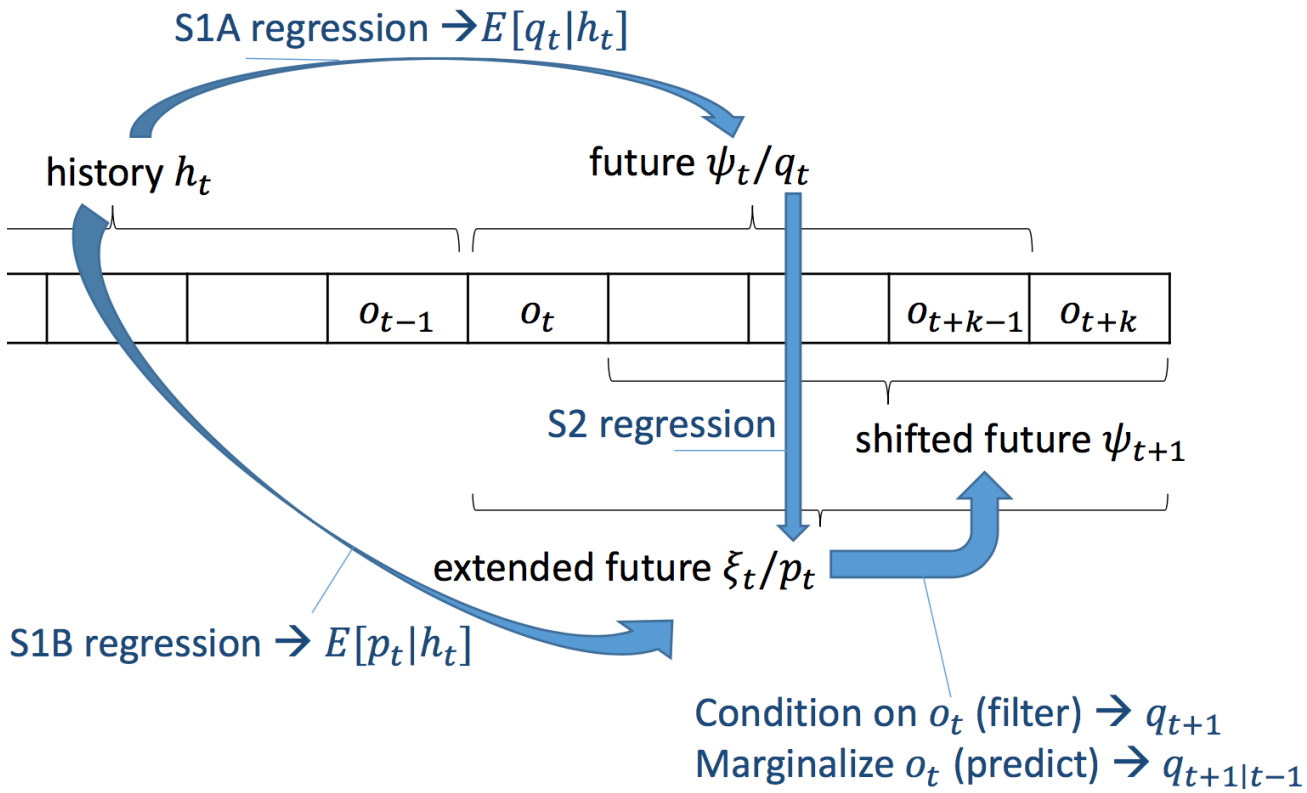


Figure 2: Predictive state representation[3].

where $\psi_t = \psi(o_{t:t+k-1})$ is future feature, $\xi_t = \xi(o_{t:t+k})$ is extended future feature and $h_t = h(o_{1:t-1})$ is history feature.

2.1. Formulation

- PSR is not popular, but it has good theoretical properties.
- Model hierarchy: HMM < PSR < RNN.
- PSR can be extend to non-linear discrete spaces and continuous space using a Hilbert Space Embedding technique.

2.2. Spectral learning algorithm

Define

- $q_t = q_{t|t-1} = \mathbb{E}[\psi_t \mid o_{1:t-1}]$ determins $\mathbb{P}(o_{t:t+k-1} \mid o_{1:t-1})$
- $p_t = p_{t|t-1} = \mathbb{E}[\xi_t \mid o_{1:t-1}]$ determins $\mathbb{P}(o_{t:t+k} \mid o_{1:t-1})$

For a linear system, $p_t = Wq_t$. we have

$$\mathbb{E}[p_t \mid h_t] = \mathbb{E}[Wq_t \mid h_t] \tag{3}$$

$$\mathbb{E}\Big[\mathbb{E}[\xi_t \mid o_{1:t-1}] \mid h_t\Big] = W\mathbb{E}\Big[\mathbb{E}[\psi_t \mid o_{1:t-1}] \mid h_t\Big] \tag{4}$$

$$\mathbb{E}[\xi_t \mid h_t] = W\mathbb{E}[\psi_t \mid h_t] \tag{5}$$

We can estimate $\hat{\mathbb{E}}[\xi_t \mid h_t], \hat{\mathbb{E}}[\psi_t \mid h_t]$ from training data, then use ridge regression to calculate $\hat{W}$. The initial state $\hat{q}_1 = \hat{\mathbb{E}}[\psi_1]$. To update states, we use $q_{t+1} = f_{filter}(p_t, o_t)$ or $q_{t+1} = f_{predict}(p_t)$.

2.3.1. Refining model: Gradient Descent

If $q_{t+1} = f_{filter}(q_t, o_t) = \frac{B_{o_t} q_t}{q_\infty^T B_{o_t} q_t}$, where q_∞ is a normalization vector.

The negative log likelihood function is

$$\begin{aligned} & L(B_1, ..., B_M) \\ = & - log(\mathbb{P}(O)) \\ = & - log(q_\infty^T B_{o_T} B_{o_{T-1}} ... B_1 q_1) \end{aligned}$$

Then, we can use gradient descent to refine this model[4].

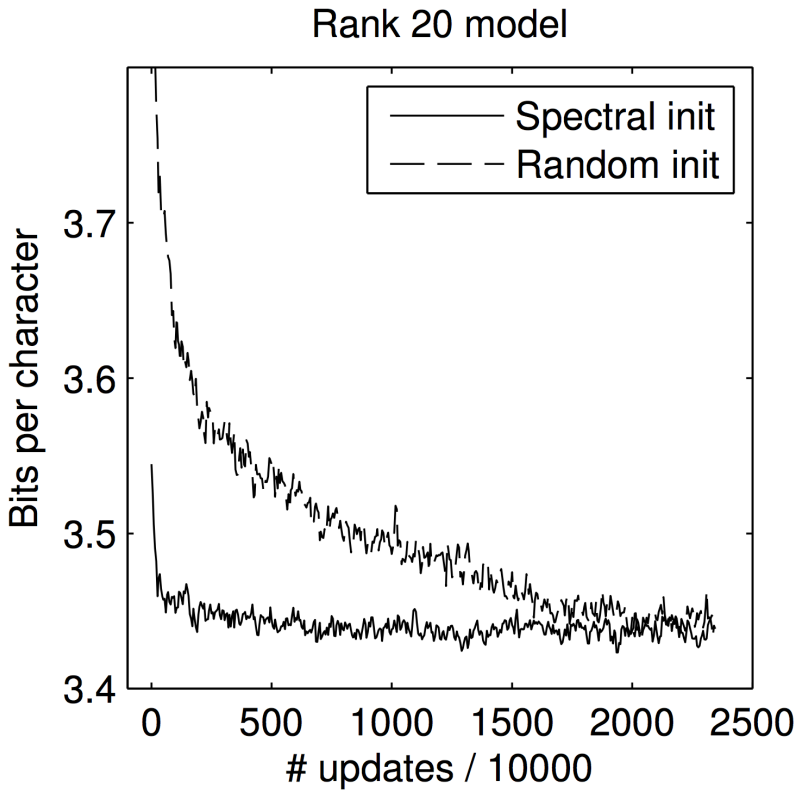


Figure 3: Character-level language model performance on Wikipedia data[4]. RNN could reach 1.5 - 1.6 bpc[6].

2.3.2. Refining model: RNN

PSR and RNN have very similar recursive structures. What if we use a RNN to refine a PSR? We can just apply spectral-learned weights to a RNN and use normal training algorithms to refine it.

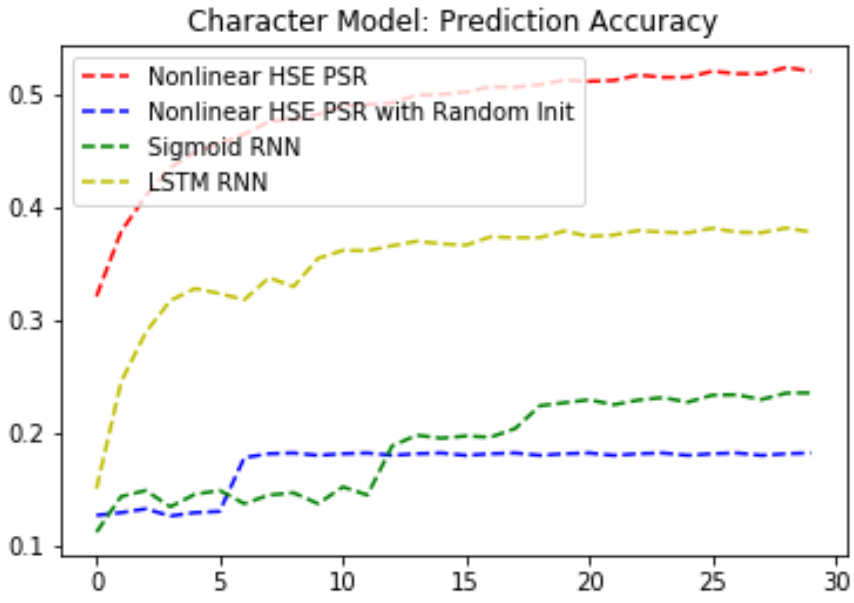


Figure 4: Character-level language model performance on PennTreebank.

Thanks!

References I


Z. Ghahramani and G. E. Hinton.
Parameter Estimation for Linear Dynamical Systems.
Technical Report, 6(CRG-TR-96-2):1–6, 1996.


Z. Ghahramani and S. T. Roweis.
Learning Nonlinear Dynamical Systems using an EM Algorithm.
Advances in Neural Information Processing Systems 11, (1):599–605, 1999.


A. Hefny, C. Downey, and G. Gordon.
Supervised Learning for Dynamical System Learning.
pages 1–32, 2015.

References II

N. Jiang, A. Kulesza, and S. Singh.
Improving Predictive State Representations via Gradient Descent.
Proceedings of the 30th Conference on Artificial Intelligence (AAAI 2016), pages 1709–1715, 2016.

J. Langford, R. Salakhutdinov, and T. Zhang.
Learning Nonlinear Dynamic Models.
Proceedings of the 26th Annual International Conference on Machine Learning ICML 09, cs.AI(1):1–8, 2009.

J. Martens.
Generating Text with Recurrent Neural Networks.
Neural Networks, 131(1):1017–1024, 2011.

Overview of Q-Learning Convergence

Vivek Nangia

Carnegie Mellon University

vnangia@andrew.cmu.edu

May 4, 2017

Introduction to Reinforcement Learning

- Agent moving around some discrete, finite world
- Choose action from finite collection of actions at every step
- At time n, agent at state $x_n \in X$ choses action $a_n \in A$
- The agent receives probabilistic reward r_n. Mean value $R_{x_n}(a_n)$ is dependent only on state and action.
- Agent moves to next state y_n with dynamics

$$Pr(y_n = y | s_n, a_n) = P_{x_n y}[a_n]$$

- Goal: Determine optimal policy that maximizes total expected discount reward

State Value Functions

- For a given policy π, the value of state x is defined as:

$$V^{\pi}(x) = R_x(\pi(x)) + \gamma \sum_y P_{xy}[\pi(x)] V^{\pi}(y)$$

- The agent expects to receive $R_x(\pi(x))$ immediately and the moves to a state that is 'worth' $V^{\pi}(y)$ with probability $P_{xy}[\pi(x)]$

Optimal State Value

- Dynamic programming theory states there exists at least one optimal policy π^* such that

$$V^*(x) = V^{\pi^*}(x) = \max_a \left(R_x(a) + \gamma \sum_y P_{xy}[a] V^{\pi^*}(y) \right)$$

- This is a well defined recursive definition with the ability determine V^* and π^* assuming $R_x(a)$ and $P_{xy}[a]$ are known.

Q Values

- For a policy π, define Q values (action-values) as:

$$Q^{\pi}(a, x) = R_x(a) + \gamma \sum_y P_{xy}[\pi(x)] V^{\pi}(y)$$

- Q value is the expected discounted reward for executing action a at state x and then following policy π

Q - Learning

- Seeks to learn Q values, state for optimal policy π^*
- $V^*(x) = \max_a Q^*(x, a)$
- This leads to the policy $\pi^*(x) = a^*$

Episodes

- Stepping through the environment consists of distance *episodes*
- The nth episodes is defined as the tuple $(x_n, a_n, y_n, r_n, \alpha_n)$
  - Current state x_n
  - Current action a_n
  - Subsequent state y_n
  - Reward r_n
  - Learning factor α_n where

$$Q_n(x, a) = (1 - \alpha_n)Q_{n-1}(x, a) + \alpha_n[r_n + \gamma V_{n-1}(y_n)]$$

Conditions on Convergence

- $Q_n(x, a)$ is represented as a lookup table
- Sequence of episodes has infinite length
- Episodes need not be continuous

Action Replay Process (ARP)

- Artificial controlled Markov Process over the sequences of episodes
- $< x, n > \rightarrow^{r_t} < y, t-1 >$, where at episodes $t < n$, go from state x to y with action a.

Lemma A

$Q_n(x, a)$ are the optimal action values for the ARP states $< x, n >$ and actions

Lemma B

- Given a finite Markov process, from any starting state x, the different between the value of that state under the finite sequences of s actions and its value under the same sequences followed by any other action tends to 0 as $n \Rightarrow \infty$
- Given l, there exists a higher level h such that the probability can be made arbitrarily small of moving below l after taking s actions in the ARP
- With probability 1, the probabilities of the ARP and reward function of the ARP tend to those values in the real process as the level n increases to infinity
- As the dynamics and expected reward functions of the ARP and real process converge, the Q values of all states converge

Proof

The ARP tends toward the real process, and so its optimal Q values do as well. But $Q_n(a, x)$ are the optimal Q values for the nth level of the ARP, and therefore tend to $Q^*(x, a)$

The End

Monte Carlo Methods for Sampling

Hengyuan Hu, Quanbin Ma

Motivation

Sampling is pervasive in statistics and machine learning.

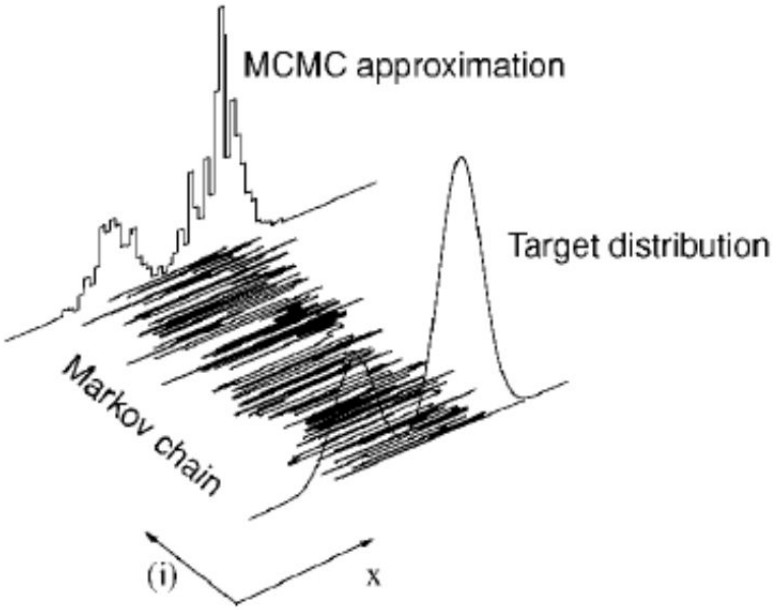
- Draw samples from arbitrary distribution.
- Use samples to estimate intractable terms.
- Extremely hard especially in high dimension.

Use Monte Carlo methods to trade accuracy for efficiency.

Markov Chain Monte Carlo for sampling

Metropolis Hastings Algorithm

- Given a state:
- **Proposal**: go to any stochastic perturbation of the state.
- A common choice is following a Gaussian: Random Walk
- **Correction**: rejects any proposals that stray too far away from the target distribution.
- Consecutive steps are highly correlated: inefficient exploration



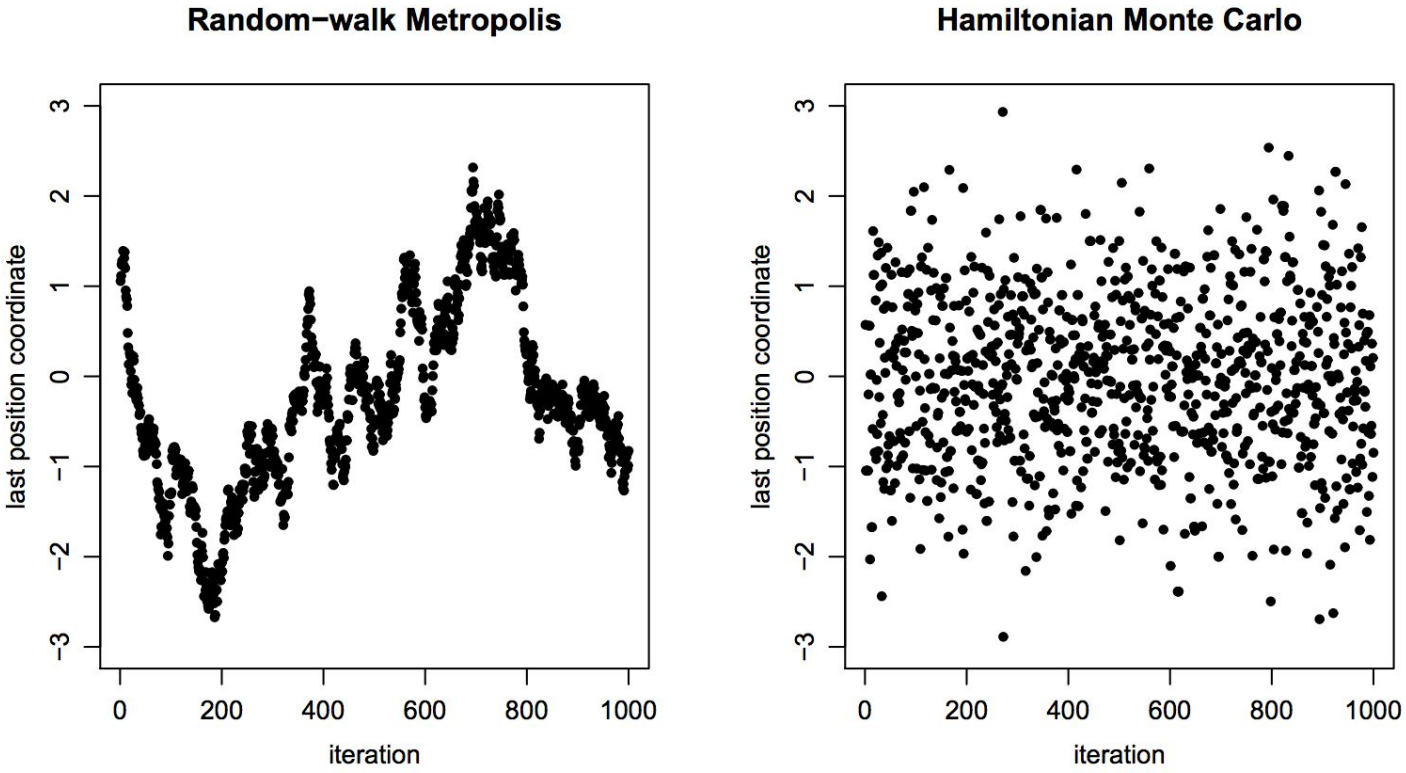
Hamiltonian Monte Carlo

- Model not only the state, but also velocity.
- Formulated under framework of Hamiltonian dynamics to ensure basic properties of MCMC are satisfied.

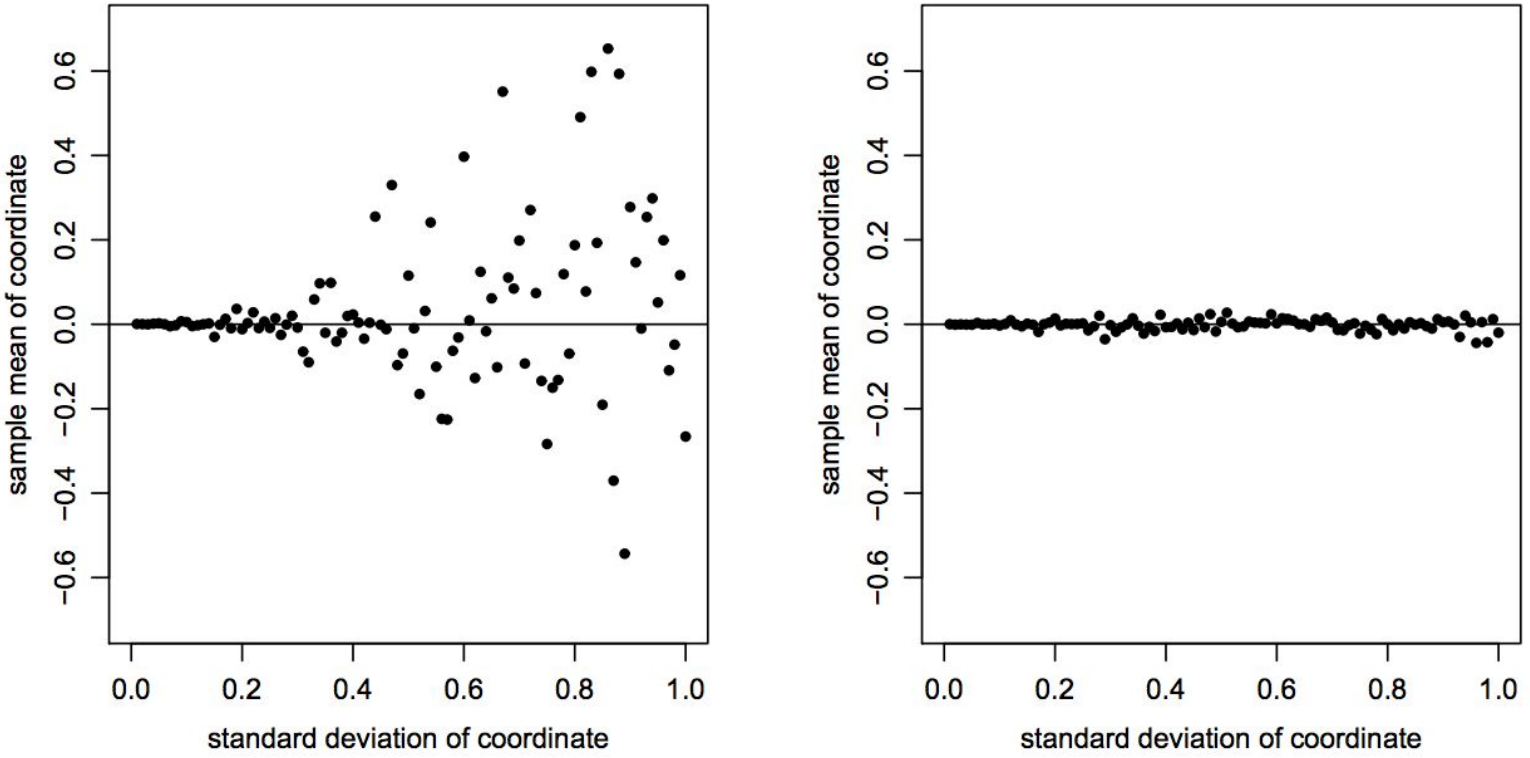
$$E(\mathbf{x}, \mathbf{v}) = U(\mathbf{x}) + K(\mathbf{v})$$

- Instead of walking, we are now kicking a ball around.

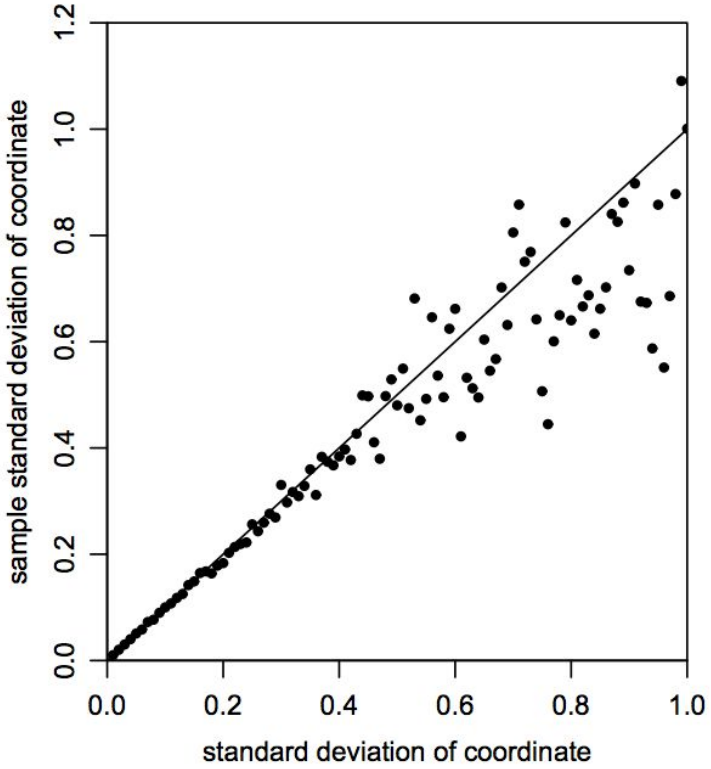
Samples generated for an 100-dimensional Gaussian. (Neal 2011)

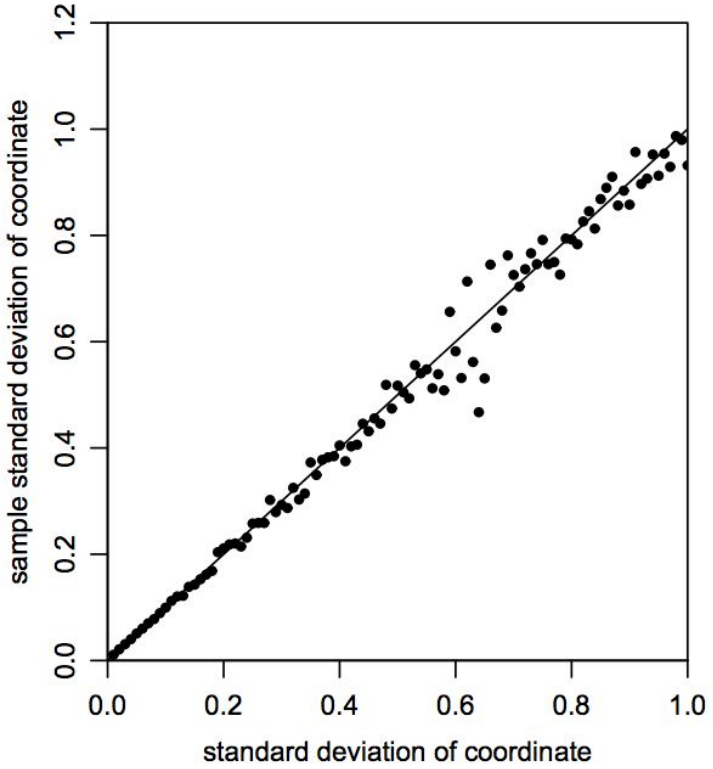


Samples generated for an 100-dimensional Gaussian. (Neal 2011)



Samples generated for an 100-dimensional Gaussian. (Neal 2011)





Probabilistic Inference in Graphical Models via Markov chain Monte Carlo Method

Yuan Liu（刘源）

Motivation

What is inference?

- Let F, E be two disjoint subsets of the random variables V. Denote $R = V - E - F$. The probabilistic inference is to calculate: $p(x_F|x_E) = \frac{\sum_{x_R} p(x_E,x_F,x_R)}{\sum_{x_R,x_F} p(x_E,x_F,x_R)}$
- The calculation of $E_{x\sim p}[f(x)]$

Why it is important?

- We always need to calculate the marginal distribution.
- Write the Markov Random Field in log-linear form: $p(x_V) = \frac{1}{Z}\exp(\prod_{c\in C} w_c F_c(x_c))$. The derivation has the following form: $\frac{\partial}{\partial w_c} l(w: D) = \sum_{x\in D} F_c(x_c) - E_{x\sim p}[F_c(x_c)]$

The Metropolis-Hastings Algorithm

This algorithm maintains a record of the current state $x^{(t)}$, and the proposal distribution $q_k(x, x^{(t)})$ depends on the current state.

$$A_k\left(x^*, x^{(t)}\right) = \min(1, \frac{p(x^*)q_k(x^*, x^{(t)})}{p\left(x^{(t)}\right)q_k(x^{(t)}, x^*)})$$

- Prove the convergence of this algorithm.
- Burning–in the sampler.
- Simulated Annealing.
- Effective sample size.

Gibbs Sampling

Gibbs sampling is a special case of the Metropolis-Hasting algorithm.
$q_k(x, x') = p(x'_k | x_{-k})$.

- The main advantage of Gibbs sampling is the new state will always be accepted.
- How to use it in the Markov Random Field. (Markov Blanket)
- Blocking Gibbs Sampling
- Convergence Diagnostics: The Gibbs Stopper.

Thank you

Asynchronous & Approximate Q-learning

Emilio Parisotto
eparisot@andrew.cmu.edu

PhD Student
Machine Learning Department
Carnegie Mellon University

Carnegie Mellon University

MACHINE LEARNING DEPARTMENT

Reinforcement Learning

- Given an MDP $\left(S, A, \gamma, P(s'|s,a), r(s,a,s')\right)$
  - Find a policy $\pi : S \rightarrow A$ that maximizes future reward.

- Q-function:
  - Expected future reward starting in state s, executing a, and following π

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

- Satisfies the Bellman Equation:

$$Q^{\pi}(s,a) = \mathbb{E}_{a' \sim \pi(\cdot|S), s' \sim P(\cdot|S,a)}[r(s,a,s') + \gamma Q^{\pi}(s',a')]$$

Q learning

- Optimal Q-function satisfies Bellman Optimality Equation:

$$Q^{\pi^*}(s,a) = \mathbb{E}_{s' \sim P(s'|s,a)} \left[r(s,a,s') + \gamma \max_{a'} Q^{\pi^*}(s',a') \right]$$

- We can extract optimal policy by choosing action with highest Q-value.
- Q-learning recovers optimal Q-value:

$$Q_{t+1}(s,a) = (1 - \gamma_t(s,a)) Q_t(s,a) + \gamma_t(s,a) \left(r(s,a,s') + \min_{a' \in \mathcal{A}} Q_t(s',a') \right)$$

Asynchronous Q-learning Theorem

Theorem 5: Let **A1** from Section 4 hold. Then, $Q_t(s,a)$ converges with probability 1 to $Q^*(s,a)$ for every s and a for each of the following cases:

1. If all policies are proper.
2. If **A3** and **A4** from Section 4 hold (there exists a proper policy and every improper policy has infinite cost for some initial state), and if $Q_t(s,a)$ is guaranteed to be bounded, with probability 1.

Meaning: If we use Q-learning update asynchronously, Q-learning still converges to optimal Q-value function with some minor assumptions on the MDP.

$$Q_{t+1}(s,a) = (1-\gamma_t(s,a))Q_t(s,a) + \gamma_t(s,a)\left(r(s,a,s') + \min_{a' \in \mathcal{A}} Q_t(s',a')\right)$$

Approximate Q-learning Theorem

Theorem 6: Let **A1** from Section 4 hold. Let π be an ergodic policy and let $\pi(s,a) > 0$ for all $s \in \mathcal{S}$, $a \in \mathcal{A}$. Let the definitions from Section 3.3 hold and assume that $\{(s_t, a_t, r_t)\}$ are sampled according to the policy π. Then the update recursion in Eq. 3 converges with probability 1 to the fixed point:

$$Q_{\theta^*}(s,a) = (\mathcal{P}HQ_{\theta^*})(s,a)$$

where $\mathcal{P}$ is the operator:

$$(\mathcal{P}Q)(s,a) = \Phi(s,a)^T \mathscr{P} Q$$

In addition, the limit function satisfies the following error bound:

$$\|Q(\theta^*) - Q^*\|_\infty \leq \frac{1}{1-\gamma} \|\mathcal{P}Q^* - Q^*\|_\infty$$

Meaning: Q-learning converges to a fixed point which is close to the best possible approximation of Q in the linear function class we are considering, given some assumptions on the features.

Thank you

Latent-Feature Model & Latent-Feature Lasso

Enxu Yan Xiaofei Shi

Problem Definition: Latent-Feature Model (LFM)

- LFM is a generalization of Mixture Model:

$$x = W^T z + \omega.$$

- $z \in \{0,1\}^K$: presence or absence of features $\{W_{k,:}\}_{k=1}^K$.
- $\omega \in \mathbb{R}^D$ contains *noise* and *bias* due to model misspecify.

Goals:

- (i) Estimate W (if model is correct).
- (ii) Minimize $r(W) := \mathbb{E}\left[\min_{z\in\{0,1\}^K} \|x - W^T z\|^2\right]$.

Existing and New Results

Let $r(W) := \mathbb{E}\left[\min_{z\in\{0,1\}^K} \|x - W^T z\|^2\right]$ be the risk.

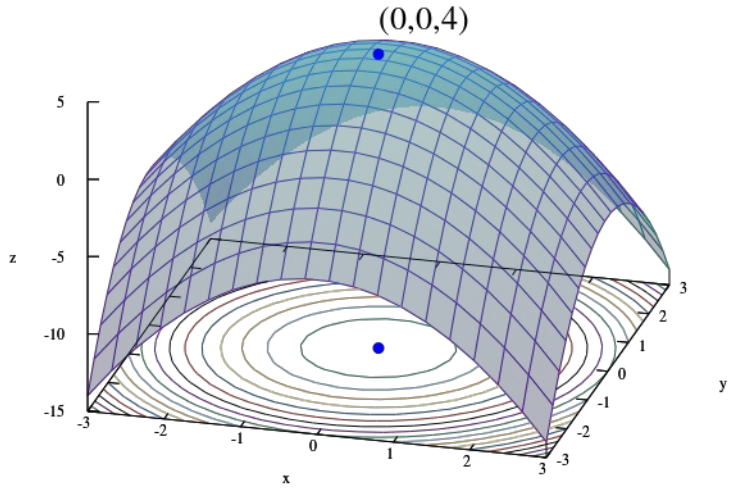
Table: Summary of Results (N=#samples, D=dim(x), K=#features).

Method	Complexity	Theoretical Guarantee	Assumption
MCMC	$(N\hat{K}^2 D)$	$(Z^{(T)}, W^{(T)}) \sim p(Z, W\|X)$ as $T \to \infty$	$Z \sim$IBP(α), $x \sim N(W^T z, \sigma^2 I)$
Varitional	$(N\hat{K}^2 D)$	n/a	n/a
MF-Binary	$(NK)2^K$	Exact Recovery	Noiseless with cond. on Z^*
MAD-Bayes	$(NK^3 D)$	Reach local minimum of $r_N(W)$	n/a
Spectral	$ND + K^5$	$\|W_j - W^*_{\tau(j)}\| \leq \epsilon, \quad N \gtrsim DK^6 \pi_{min}^{-6} \epsilon^{-2}$	$z \sim$Bern(π), $x \sim N(W^{*T} z, \sigma^2 I)$
LF Lasso (New)	$(ND + \hat{K}^2 D)$	(i) $r(W) - r(W^*) \leq \epsilon, \quad N \gtrsim DK\epsilon^{-3}$ (ii) Exact Recovery	(i) Any $p(x)$ with bounded domain. (ii) Noiseless with condition on Z^*

- LFLasso enjoys (i) no assumption on $p(x)$ except boundedness, (ii) sample complexity linear to K and (iii) a lower computational complexity.

Family of Variance Reduction Methods in Stochastic Gradient Descent

Xiyu Wang



problem to minimize finite sums:

$minP(\omega), \mathrm{P}(\omega) = \frac{1}{n}\sum_{i=1}^{n} f_i(\omega)$

SGD:

- can obtain an $O\left(\frac{1}{\sqrt{T}}\right)$ convergence rate in expectation;
- Large variance can slow down the convergence

SAG:

- introduces a convex combination of SGD direction samples, and a past stored gradient
- has a biased update direction
- stores all the gradients during calculation

SVRG:

- The update rule is:
  - $\omega^t = \omega^{t-1} - \eta\left(\nabla f_{i_t}(\omega^{t-1}) - \nabla f_{i_t}(\tilde{\omega}) + \nabla P(\tilde{\omega})\right)$
- The learning rate of SVRG can reach the result of SAG, but without storing a table of all the gradients.
- Under mild assumptions, even work on non-convex cases
- The geometric convergence is:
  - $E[P(\tilde{\omega}_s) - P(\omega_*)] \leq \alpha^s[P(\tilde{\omega}_0) - P(\omega_*)]$

SAGA:

- Combines the benefit of SVRG and SAG.
- The update rule is:
  - $\omega^t = \omega^{t-1} - \eta\left(\nabla f_i(\omega^t) - \nabla f_i\left(\emptyset_i^t\right) + \frac{1}{n}\sum_{i=1}^n \nabla f_i\left(\emptyset_i^t\right)\right)$
- Does not require the additional step to choose the number of inner loops
- If f_i is strongly convex, the convergence is:

$$E\left\|\omega^{(t)} - \omega_*\right\|_2^2 \leq \left(1 - \frac{\mu}{2(\mu n + L)}\right)^t \left[\left\|\omega^{(0)} - \omega_*\right\|_2^2 + \frac{n}{\mu n + L}[P\left(\omega^k\right) - \langle \nabla P(\omega_*), \omega^{(0)} - \omega_* \rangle - P(\omega_*)]\right] \quad (12)$$