36-724 Spring 2006: EDA, Prediction, Classification, Modeling

Brian Junker

March 22A, 2006

• Choice of Loss in the Expected Prediction Error (EPE)
• Discrete Classification and the Bayes Classifier
• The Curse of Dimensionality
• MSE and the Bias-Variance Tradeoff
• $K$-fold Cross-Validation
• Directions in which to extend linear regression and $k$-nn methods
Choice of Loss in the Expected Prediction Error (EPE)

Let $Y$ be a random variable representing outcomes and $X = (X_1, \ldots, X_p)^T$ be a random vector of predictors (covariates). We wish to construct a function $f(x)$ such that $Y \approx f(X)$ over the joint distribution $P(dx, dy)$.

We previously considered

$$EPE(f) = E[L(Y, f(X))] = \int (y - f(x))^2 P(dx, dy)$$

for squared error loss $L[Y, f(X)] = (Y - f(X))^2$, and concluded $f(x) = \text{argmin}_c E_{Y|X}[(Y - c)^2 | X = x] = E[Y|X = x]$

- For $L_1$ loss $L[Y, f(X)] = |(Y - f(X)|$, minimizing $EPE$ leads to $f(x) = \text{median}(Y|X = x)$.

- In discrete classification problems with $K$ categories, $Y \in \{1, \ldots, K\}$, may write the loss as a $K \times K$ matrix $L = [L(k, \ell)]$, where $L(k, \ell)$ is the loss in misclassifying a category-$k$ object into category $\ell$. 
Discrete Classification and the Bayes Classifier

For the discrete loss matrix $L = [L(k, \ell)]$,

$$EPE = E[L(Y, f(X))] = E_X \left\{ \sum_{k=1}^{K} L(k, f(X)) P(Y = k|X) \right\}$$

and again we can minimize pointwise to obtain

$$f(x) = \arg\min_{\ell} \sum_{k=1}^{K} L(k, \ell) P(Y = k|X = x)$$

When $L(k, \ell) = 1_{\{k \neq \ell\}}$, zero-one loss, we obtain

$$f(x) = \arg\min_{\ell} \sum_{k \neq \ell} P(Y = k|X = x)$$

$$= \arg\min_{\ell} \{1 - p(Y = \ell|X = x)\}$$

$$= \arg\max_{\ell} P(Y = \ell|X = x)$$

This the the posterior mode, or Bayes classifier. Its error rate is the Bayes rate.
**k-nearest-neighbors and the linear classifier**

In a two class problem... 

- The k-nn classifier averages class assignment over the k nearest neighbors to \( x \), and assigns to the category that gets the “majority vote”. Thus it approximates

\[
\arg\max_{\ell} P(Y = \ell|X = x) \approx \text{Highest-frequency category } \ell \text{ among } k \text{ nearest neighbors to } x
\]

- The linear regression approach can similarly be viewed as
  - Fit the linear model \( \hat{P}[Y = k|X = x] = \hat{E}[Y = k|X = x] = x^T \hat{\beta} \);
  - Assign according to the largest value of \( \hat{P}[Y = 1|X = x] \).

Viewed in this way, both methods are modeling \( P[Y = k|X = x] \) and both models suggest natural generalizations to more than \( K = 2 \) categories.
The Curse of Dimensionality

Consider $k$ nearest neighbor method applied to data uniformly distributed in the unit cube in $\mathbb{R}^p$.

- To capture a fraction $r$ of the unit volume (analogous to $k$-nn for a fixed sample size) we need to encompass a cube of edge length $r^{1/p}$. As $p \to \infty$, $r^{1/p} \to 1$!

- Clearly data gets “farther apart” as $p$ grows!

- Similarly, the sampling density in any small patch of the unit cube is proportional to $N^{1/p}$, so the SE for a local estimate of a mean or proportion “at a fixed distance” from $x$ decreases like $N^{-1/2p}$.

- If $N = 100$ is sufficient to estimate a local mean in $\mathbb{R}^1$, need $N = 100^{10}$ observations to estimate the corresponding local mean in $\mathbb{R}^{10}$!
MSE and the Bias-Variance Tradeoff

Let $\mathcal{T} = ((x_1, y_1), \ldots, (x_n, y_n))$ be a training set on which we train our predictor $\hat{y}_i = \hat{f}(x_i)$. Let $y_0 = f(x_0)$ represent a new pair, or test point. We can write the mean-squared error in predicting $y_0 \approx \hat{y}_0 = \hat{f}(x_0)$ as

$$MSE(x_0)$$

$$= E_T [(f(x_0) - \hat{y}_0)^2$$

$$= E_T [\hat{y}_0 - E_T (\hat{y}_0)]^2 + [E_T (\hat{y}_0) - f(x_0)]^2$$

$$= \text{Var}_T (\hat{y}_0) + \text{Bias}^2$$

This is an average over the distribution of training sets, holding the test point fixed.
• For 1-nn:
  
  – For \( N \) large and \( p \) small, both terms can be small for 1-nn, e.g.
  
  – As \( p \) grows, neighbors get farther apart, and both terms can grow.
  
  – Certainly the variance grows due to the \( N^{-1/2p} \) behavior mentioned above.

• For least squares regression \( Y = X\beta + \epsilon \):

\[
MS\ E(x_0) = EPE(x_0) = \sigma^2 + E_{x_0} x_0^T (X^T X)^{-1} x_0 \sigma^2
\]

\[
E_{x_0} MSE(x_0) = \int MSE(x_0) p(x_0) dx_0 = \sigma^2 + (p/N) \sigma^2
\]

– So by assuming a strong model we can reduce the effects of the curse of dimensionality [speeding the rate of \( MSE \to 0 \), from \( O(N^{-1/2p}) \) to \( O(N^{-1}) \)].
What accounts for the difference?

- \( K \text{-nn} \) doesn’t impose any smoothness across the data space; it depends entirely on \textit{local averages}. As \( p \) grows, there is very little local information from which to compute local averages.

- LS regression imposes smoothness across the data space, and relies on global averages. Thus it doesn’t rely on filling any local neighborhoods with sample points.
Note that

\[ EMS\ E = \int_{x_0 \in \mathcal{D}} MSE(x_0)p(x_0)dx_0 = \int_{x_0 \in \mathcal{D}} E_{\mathcal{T}}[f(x_0) - \hat{y}_0]^2 p(x_0)dx_0 \]

\[ = E_{\mathcal{T}} \int_{x_0 \in \mathcal{D}} [f(x_0) - \hat{y}_0]^2 p(x_0)dx_0 \]

is a double average over training sets \( \mathcal{T} \) and test points \( x_o \in \mathcal{D} \), the data space.

A typical estimate is to split the data into a training set \( \mathcal{T} \) and a test set \( \mathcal{S} \), fit \( \hat{y} = \hat{f}(x) \) on \( \mathcal{T} \), and then compute the sample MSE on \( \mathcal{S} \).

This

- Replaces the \( E_{\mathcal{T}} \) with a plug-in for the particular \( \mathcal{T} \) we have to work with.
- Replaces integrating over the whole data space \( \mathcal{D} \) with averaging over the test set \( \mathcal{S} \).
**K-fold Cross-Validation**

- We can see from the (E)MSE plot on the previous page that the error on the training data is likely to be too optimistic.
- That calculation was based on dividing data into large “training” and “testing” samples. Train on one sample, estimate the error rate on the other.
- A better estimate of EMSE would average over several training and test sets. But data is scarce! An efficient way to re-use a single data set for this purpose is *K-fold cross-validation*:
  - Divide up the data into $K$ roughly-equal-sized parts.
  - Let $\hat{f}(x)^{-k}$ be the fitted value (classification, prediction, etc.) for $x$ with the $k^{th}$ part of the data removed, and let $k(i)$ be the part of the data containing $x_i$.
  - Then the $K$-fold cross-validation criterion is
    $\quad CV = \frac{1}{N} \sum_{i=1}^{N} L(y, \hat{f}^{-k(i)}(x_i))$
  - where $L(y, \hat{y})$ is some appropriate loss function [e.g. $L(y, \hat{y}) = (y - \hat{y})^2$, if we are interested in (E)MSE].
Choosing the number $K$ of cross validation groups

CV can also be interpreted as trying to estimate the true error rate of the optimal $f(x)_{opt}$:

$$\text{Error}(f_{opt}(\cdot)) \approx \frac{1}{K} \sum_{k=1}^{K} \text{Error}(\hat{f}^{-k}(\cdot)) \approx \frac{1}{N} \sum_{k=1}^{N} L(y, \hat{f}^{-k(i)}(x_i)) = CV$$

- There is also a bias/variance tradeoff in CV estimates of prediction error:
  - $K$ large: Less bias for the true prediction error; but potentially high variance because the $N$ training sets are so similar to each other (hence different $\hat{f}$’s are highly dependent).
  - $K$ small: tends to reverse these effects (smaller variance, larger bias).

Often the point of diminishing returns in this bias/variance tradeoff is around $K = 5$ or 10.

- $K = N$ is called “leave-one-out” cross-validation. $\hat{f}^{-k}(x)$ least biased for $f_{opt}(x)$ (largest training sets), but most variance over replicated data sets (the training data sets are essentially all identical).
Using Cross-Validation

- In principle, let $\alpha$ be any parameters of interest in $f(x; \alpha)$, and define
  \[ CV_K(\alpha) = \frac{1}{N} \sum_{k=1}^{N} L(y, \hat{f}^{-k(i)}(x_i; \alpha)) \]

- Can use cross-validation to estimate $\hat{\alpha}$ (e.g. gradient-descent, etc.). This is sometimes done, but it is computationally expensive!

- Cross-validation is often useful for “tuning parameters” that are not part of the main model fitting but have to be set in some way
  - The list of variables/features to include in the model
  - The roughness penalty in a penalized function estimation problem
  - The degree of the polynomial in polynomial regression
  - The number of neighbors in $k$-nn
  - Etc.
Aside...

- For many linear smoothers ($\hat{y} = Sy$)

\[
CV_N = \frac{1}{N} \sum_{i=1}^{N} [y_i - \hat{f}^{-i}(x_i)]^2 = \frac{1}{N} \sum_{i=1}^{N} \left[ \frac{y_i - \hat{f}(x_i)}{1 - S_{ii}} \right]^2
\approx \frac{1}{N} \sum_{i=1}^{N} \left[ \frac{y_i - \hat{f}(x_i)}{1 - \text{trace}(S)/N} \right]^2 \equiv GCV
\]

This is generalized cross-validation; it can be faster than CV, and it tends to smooth a bit more (choose models with lower $df=\text{trace}(S)$) than CV.

- Other penalized methods based on $df=\text{trace}(S)$ are also used (AIC, BIC, ...!)
Directions in which to extend linear regression and $k$-nn methods

- Linear regression needs to relax the assumptions made about $E[Y|X = x]$. This leads to
  
  - Generalized linear models, nonlinear regression
  - Penalized regression methods, spline-based methods
  - generalized additive models; Projection Pursuit regression; neural networks
  - Basis functions and dictionary methods
  - Discriminant analysis methods
  - Etc.
• $k$-nn needs to strengthen the assumptions made about $E[Y|X = x]$. This leads to
  
  – Kernel-weighting the $k$ nearest neighbors according to distance from $x$.
  
  – Fully-kernel-based methods (kernel density estimation; kernel regression methods; gaussian mixture models, etc.)
  
  – Support vector methods
  
  – Etc.