Classification 3: Logistic regression (continued); model-free classification

Ryan Tibshirani
Data Mining: 36-462/36-662

April 9 2013

Optional reading: ISL 4.3, ESL 4.4; ESL 13.1–13.3

Reminder: logistic regression

Last lecture, we learned logistic regression, which assumes that the log odds is linear in $x \in \mathbb{R}^p$:

$$\log\left\{\frac{\mathrm{P}(C=1|X=x)}{\mathrm{P}(C=2|X=x)}\right\} = \beta_0 + \beta^T x$$

This is equivalent to

$$\mathrm{P}(C=1|X=x) = \frac{\exp(\beta_0+\beta^T x)}{1+\exp(\beta_0+\beta^T x)}$$

Given a sample (x_i, y_i), $i = 1, \ldots n$, we fit the coefficients by maximizing the log likelihood

$$\hat{\beta}_0, \hat{\beta} = \underset{\beta_0\in\mathbb{R},\, \beta\in\mathbb{R}^p}{\operatorname{argmax}} \sum_{i=1}^{n} \left\{ u_i \cdot (\beta_0+\beta^T x_i) - \log\left(1+\exp(\beta_0+\beta^T x_i)\right)\right\}$$

where $u_i = 1$ if $y_i = 1$ and $u_i = 0$ if $y_i = 2$ (indicator for class 1)

Classification by logistic regression

After computing $\hat{\beta}_0, \hat{\beta}$, classification of an input $x \in \mathbb{R}^p$ is given by

$$\hat{f}^{\mathrm{LR}}(x) = \begin{cases} 1 & \text{if } \hat{\beta}_0 + \hat{\beta}^T x > 0 \\ 2 & \text{if } \hat{\beta}_0 + \hat{\beta}^T x \leq 0 \end{cases}$$

Why? Recall that the log odds between class 1 and class 2 is modeled as

$$\log\left\{\frac{\mathrm{P}(C = 1|X = x)}{\mathrm{P}(C = 2|X = x)}\right\} = \beta_0 + \beta^T x$$

Therefore the decision boundary between classes 1 and 2 is the set of all $x \in \mathbb{R}^p$ such that

$$\hat{\beta}_0 + \hat{\beta}^T x = 0$$

This is a $(p-1)$-dimensional affine subspace of $\mathbb{R}^p$; i.e., this is a point (threshold) in $\mathbb{R}^1$, or a line in $\mathbb{R}^2$

Interpretation of logistic regression coefficients

How do we interpret coefficients in a logistic regression? Similar to our interpretation for linear regression. Let

$$\Omega = \log\left\{\frac{\mathrm{P}(C=1|X=x)}{\mathrm{P}(C=2|X=x)}\right\}$$

be the log odds. Then logistic regression provides the estimate

$$\hat{\Omega} \;=\; \hat{\beta}^T x \;=\; \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \ldots + \hat{\beta}_p x_p$$

Hence, the proper interpretation of $\hat{\beta}_j$: increasing the jth predictor x_j by 1 unit, and keeping all other predictors fixed, increases

- The estimated log odds of class 1 by an additive factor of $\hat{\beta}_j$
- The estimated odds of class 1 by a multiplicative factor of $e^{\hat{\beta}_j}$

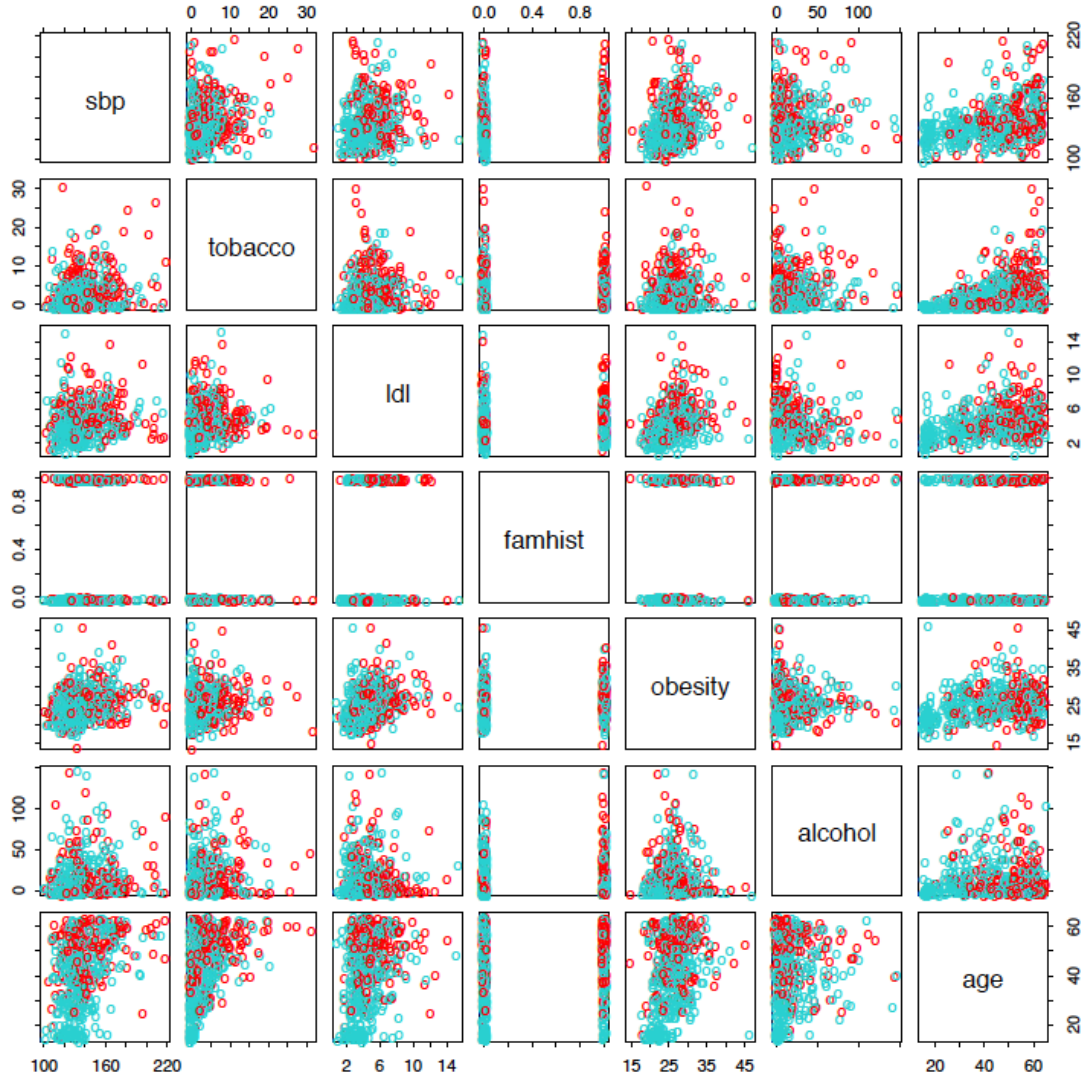
(Note: it may not always be possible for a predictor x_j to increase with the other predictors fixed!)

Example: South African heart disease data

Example (from ESL section 4.4.2): there are $n = 462$ individuals broken up into 160 cases (those who have coronary heart disease) and 302 controls (those who don't). There are $p = 7$ variables measured on each individual:

- sbp (systolic blood pressure)
- tobacco (lifetime tobacco consumption in kg)
- ldl (low density lipoprotein cholesterol)
- famhist (family history of heart disease, present or absent)
- obesity
- alcohol
- age

Pairs plot (red are cases, green are controls):



Fitted logistic regression model:

	Coefficient	Std. Error	Z Score
(Intercept)	−4.130	0.964	−4.285
sbp	0.006	0.006	1.023
tobacco	0.080	0.026	3.034
ldl	0.185	0.057	3.219
famhist	0.939	0.225	4.178
obesity	-0.035	0.029	−1.187
alcohol	0.001	0.004	0.136
age	0.043	0.010	4.184

The Z score is the coefficient divided by its standard error. There is a test for significance called the Wald test

Just as in linear regression, correlated variables can cause problems with interpretation. E.g., sbp and obseity are not significant, and obesity has a negative sign! (Marginally, these are both significant and have positive signs)

After repeatedly dropping the least significant variable and refitting:

	Coefficient	Std. Error	Z score
`(Intercept)`	-4.204	0.498	-8.45
`tobacco`	0.081	0.026	3.16
`ldl`	0.168	0.054	3.09
`famhist`	0.924	0.223	4.14
`age`	0.044	0.010	4.52

This procedure was stopped when all variables were significant

E.g., interpretation of tobacco coefficient: increasing the tobacco usage over the course of one's lifetime by 1kg (and keeping all other variables fixed) multiplies the estimated odds of coronary heart disease by $\exp(0.081) \approx 1.084$, or in other words, increases the odds by 8.4%

LDA versus logistic regression

As we remarked earlier, both LDA and logistic regression model the log odds as a linear function of the predictors $x \in \mathbb{R}^p$

$$\text{Linear discriminant analysis:} \quad \log\left\{\frac{\mathrm{P}(C=1|X=x)}{\mathrm{P}(C=2|X=x)}\right\} = \alpha_0 + \alpha^T x$$

$$\text{Logistic regression:} \quad \log\left\{\frac{\mathrm{P}(C=1|X=x)}{\mathrm{P}(C=2|X=x)}\right\} = \beta_0 + \beta^T x$$

where for LDA we form $\hat{\alpha}_0, \hat{\alpha}$ based on estimates $\hat{\pi}_j, \hat{\mu}_j, \hat{\Sigma}$ (easy!), and for logistic regression we estimate $\hat{\beta}_0, \hat{\beta}$ directly based on maximum likelihood (harder)

This is what leads to linear decision boundaries for each method

Careful inspection (or simply comparing them in R) shows that the estimates $\hat{\alpha}_0, \hat{\beta}_0$ and $\hat{\alpha}, \hat{\beta}$ are different. So how do they compare?

Generally speaking, logistic regression is more flexible because it doesn't assume anything about the distribution of X. LDA assumes that X is normally distributed within each class, so that its marginal distribution is a mixture of normal distributions, hence still normal:

$$X \sim \sum_{j=1}^{K} \pi_j N(\mu_j, \Sigma)$$

This means that logistic regression is more robust to situations in which the class conditional densities are not normal (and outliers

On the other side, if the true class conditional densities are normal, or close to it, LDA will be more efficient, meaning that for logistic regression to perform comparably it will need more data

In practice they tend to perform similarly in a variety of situations (as claimed by the ESL book on page 128)

Extensions

Logistic regression can be adapted for multiple classes, $K > 2$. We model the log odds of each class to a base class, say, the last one:

$$\log\left\{\frac{\mathrm{P}(C = j|X = x)}{\mathrm{P}(C = K|X = x)}\right\} = \beta_{0,j} + \beta_j^T x$$

We fit the coefficients, $j = 1, \ldots K$, jointly by maximum likelihood

For high-dimensional problems with $p > n$, we run into problems with LDA in estimating the $p \times p$ covariance matrix Σ: with only n observations, out estimate $\hat{\Sigma}$ will not be invertible. Regularized LDA remedies this by shrinking $\hat{\Sigma}$ towards the identity, using the estimate $c \cdot \hat{\Sigma} + (1 - c) \cdot I$ for some $0 < c < 1$

For regularization in logistic regression, we can subtract a penalty, e.g., an ℓ_1 penalty: $\lambda\|\beta\|_1$, from the maximum likelihood criterion, similar to what we did in linear regression

Model-free classification

It is possible to perform classification in a model-free sense, i.e., without writing down any assumptions concerning the distribution that generated the data

The downside: these methods are essentially a black box for classification, in that they typically don't provide any insight into how the predictors and the response are related

The upside: they can work well for prediction in a wide variety of situations, since they don't make any real assumptions

These procedures also typically have tuning parameters that need to be properly tuned in order for them to work well (for this we can use cross-validation)

Classification by k-nearest-neighbors

Perhaps the simplest prediction rule, given labeled data (x_i, y_i), $i = 1 \ldots n$, is to predict an input $x \in \mathbb{R}^p$ according to its nearest-neighbor:

$$\hat{f}^{1-\mathrm{NN}}(x) = y_i \quad \text{such that } \|x_i - x\|_2 \text{ is smallest}$$

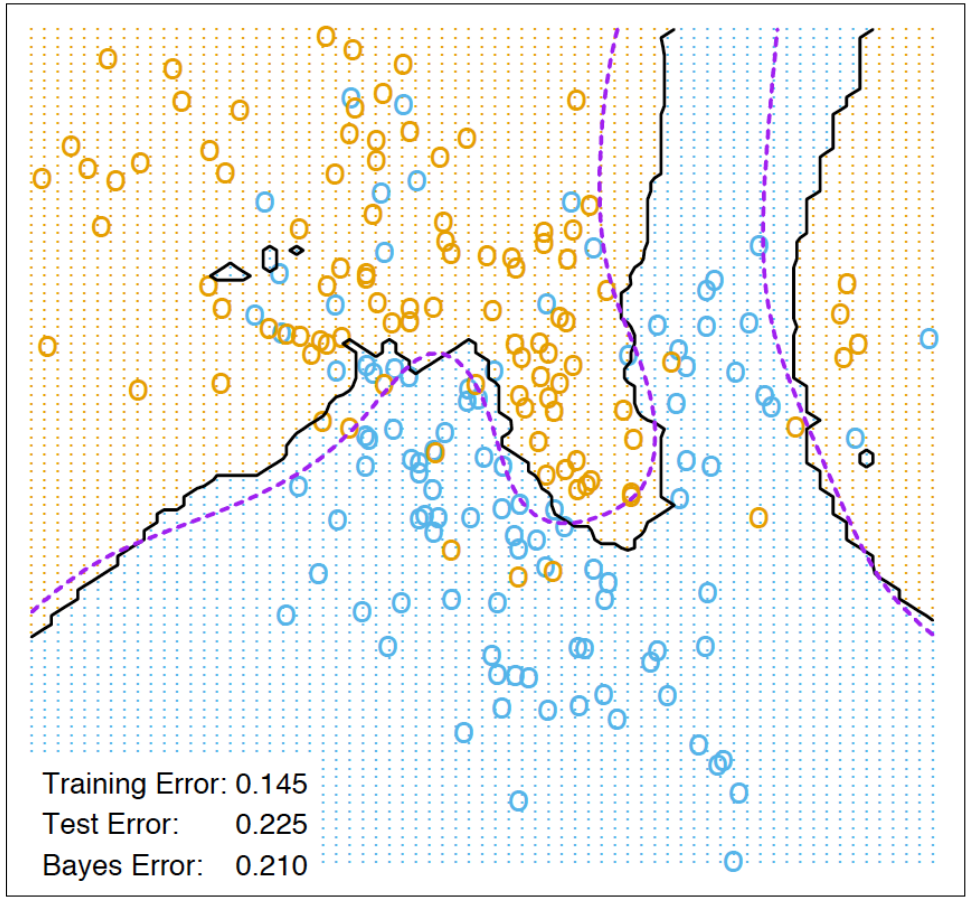
A natural extension is to consider the k-nearest-neighbors of x, call them $x_{(1)}, \ldots x_{(k)}$, and then classify according to a majority vote:

$$\hat{f}^{k-\mathrm{NN}}(x) = j \quad \text{such that } \sum_{i=1}^{k} 1\{y_{(i)} = j\} \text{ is largest}$$

What is more adaptive, 1-nearest-neighbor or 9-nearest-neighbors? What is the n-nearest-neighbors rule?

Example: 7-nearest-neighbors classifier

Example: classification with 7-nearest-neighbors (ESL page 467):



Could linear discriminant analysis or logistic regression have drawn decision boundaries close to the Bayes boundary?

Disadvantages of k-nearest-neighbors

Besides the fact that it yields limited insight into the relationship between the predictors and classes, there are disadvantages of using a k-nearest-neighbors rule having to do with computation

For one, we need the entire data set (x_i, y_i), $i = 1, \ldots n$ whenever we want to classify a new point $x \in \mathbb{R}^p$. This could end up being very prohibitive, especially if n and/or p are large. On the other hand, for prediction with LDA or logistic regression, we only need the linear coefficients that go into the prediction rule

Even with the entire data set at hand, the prediction rule is slow. It essentially requires comparing distances to every point in the training set. There are somewhat fancy ways of storing the data to make this happen as fast a possible, but they're still pretty slow

Classification by K-means clustering

Instead of using every point in the data set (x_i, y_i), $i = 1, \ldots n$, we can try to summarize this data set, and use this for classification

How would we do this? We've covered many unsupervised learning techniques; one of the first was K-means clustering. Consider the following procedure for classification:

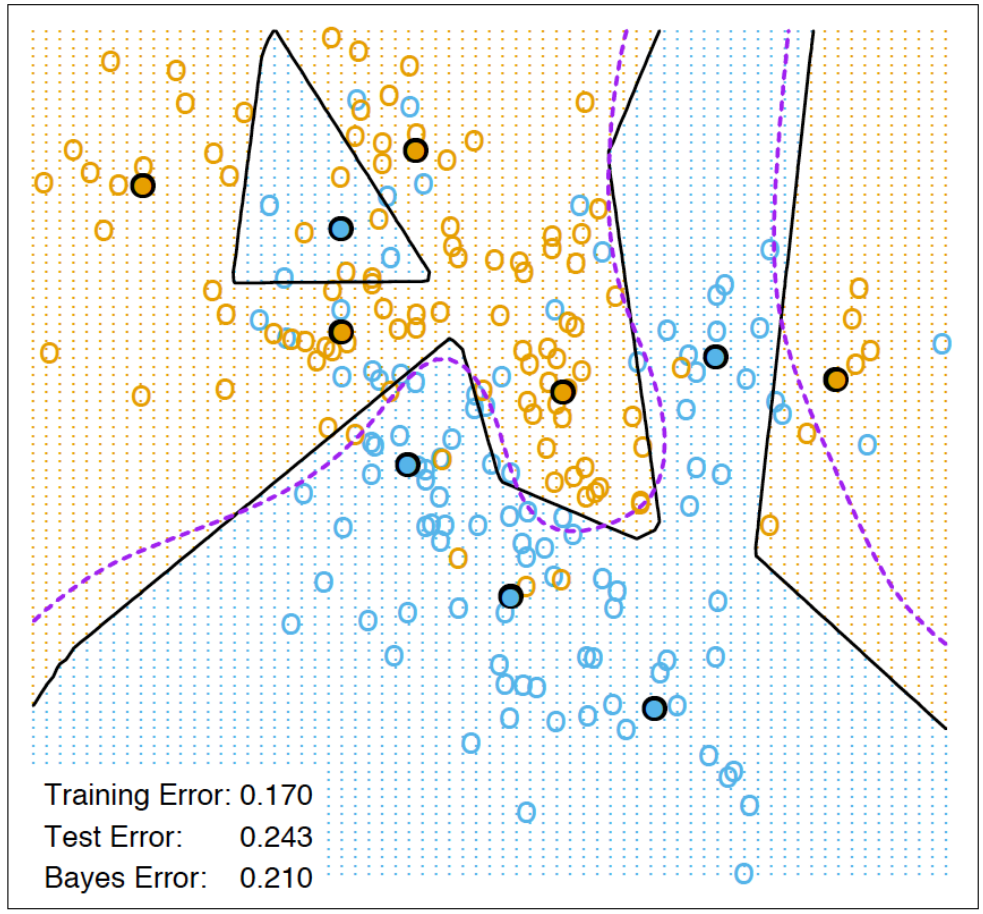
- Use R-means clustering to fit R centroids $c_1(j), \ldots c_R(j)$ separately to the data within each class $j = 1, \ldots K$;
- Given an input $x \in \mathbb{R}^p$, classify according to the class of the nearest centroid:

$$\hat{f}^{R-\text{means}}(x) = j \quad \text{such that } \|x - c_i(j)\| \text{ is smallest for some } i = 1, \ldots R$$

We only need the $K \cdot R$ centroids $c_1(j), \ldots c_K(j)$, $j = 1, \ldots K$ for classification

Example: 5-means clustering for classification

Example: same data set as before, now we use 5-means clustering within each class, and classify according to the closest centroid (from ESL page 464):



Learning vector quantization

One downside of K-means clustering used in this context is that, for each class, the centroids are chosen without any say from other classes. This can result in centroids being chosen close to decision boundaries, which is bad

Learning vector quantization[1] takes this into consideration, and places R prototypes for each class strategically away from decision boundaries. Once these prototypes have been chosen, classification proceeds in the same way as before (according to the class of the nearest prototype)

Given (x_i, y_i), $i = 1, \ldots n$, learning vector quantization chooses the prototypes as follows:

1. Choose R initial protoypes $c_1(j), \ldots c_R(j)$ for $j = 1, \ldots K$ (e.g., sample R training points at random for each class)

[1] Kohonen (1989), "Self-Organization and Associative Memory"

2. Choose a training point x_ℓ at random. Let $c_i(j)$ be the closest protoype to x_ℓ. If:
   (a) $y_\ell = j$, then move $c_i(j)$ closer to x_ℓ:

$$c_i(j) \leftarrow c_i(j) + \epsilon(x_\ell - c_i(j))$$

   (b) $y_\ell \neq j$, then move $c_i(j)$ away from x_ℓ:

$$c_i(j) \leftarrow c_i(j) - \epsilon(x_\ell - c_i(j))$$

3. Repeat step 2, decreasing $\epsilon \to 0$ with each iteration

The quantity $\epsilon \in (0, 1)$ above is called the "learning rate". It can be taken, e.g., as $\epsilon = 1/r$, where r is the iteration number

Learning vector quantization can sometimes perform better than classification by K-means clustering, but other times they perform very similarly (e.g., in the previous example)

Tuning parameters

Each one of these methods exhbitis a tuning parameter that needs to be chosen. E.g.,

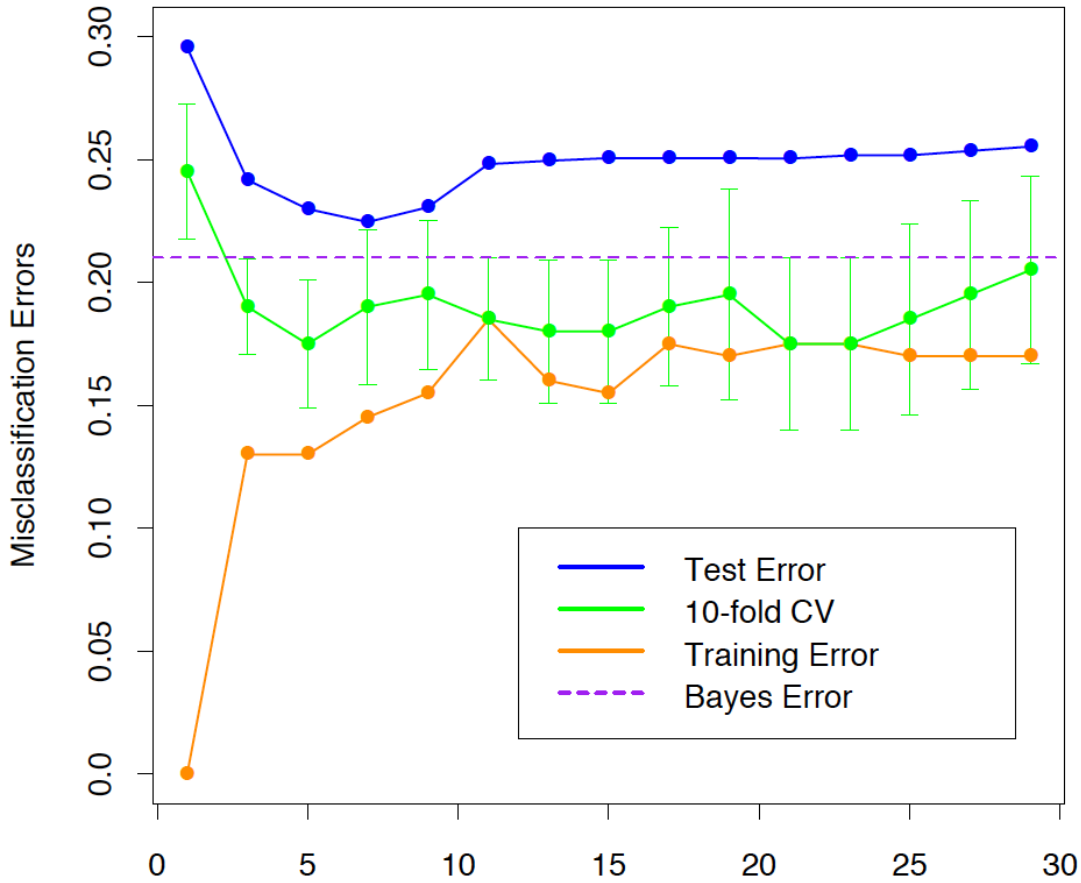
- the number k of neighbors in k-nearest-neighbors
- the number R of centroids in R-means clustering
- the number R of prototypes in learning vector quantization

Think about the bias-variance tradeoff at play here—for each method, which direction means higher model complexity?

As before, a good method for choosing tuning parameter values is cross-validation, granted that we're looking to minimize prediction error

Example: cross-validation for k-nearest-neighbors

Example: choosing the number of neighbors k using 10-fold cross-validation (from ESL page 467):



Classification tools in R

The classification tools we've learned so far, in R:

- Linear discriminant analysis: use the `lda` function in the `MASS` package. For reduced-rank LDA of dimension L, take the first L columns of the `scaling` matrix
- Logistic regression: use the `glm` function in the base package. It takes the same syntax as the `lm` function; make sure to set `family="binomial"`. Can also apply regularization using `glmnet`
- k-nearest-neighbors: use the `knn` function in the `class` package
- K-means clustering: use the `kmeans` function in the `stats` package for the clustering; classification to the nearest centroid can then be done manually
- Learning vector quantization: use the `lvq1` function in the `class` package. Here the list of protoypes is called the "codebook" and passed as the `codebk` argument

Recap: logistic regression, model-free classification

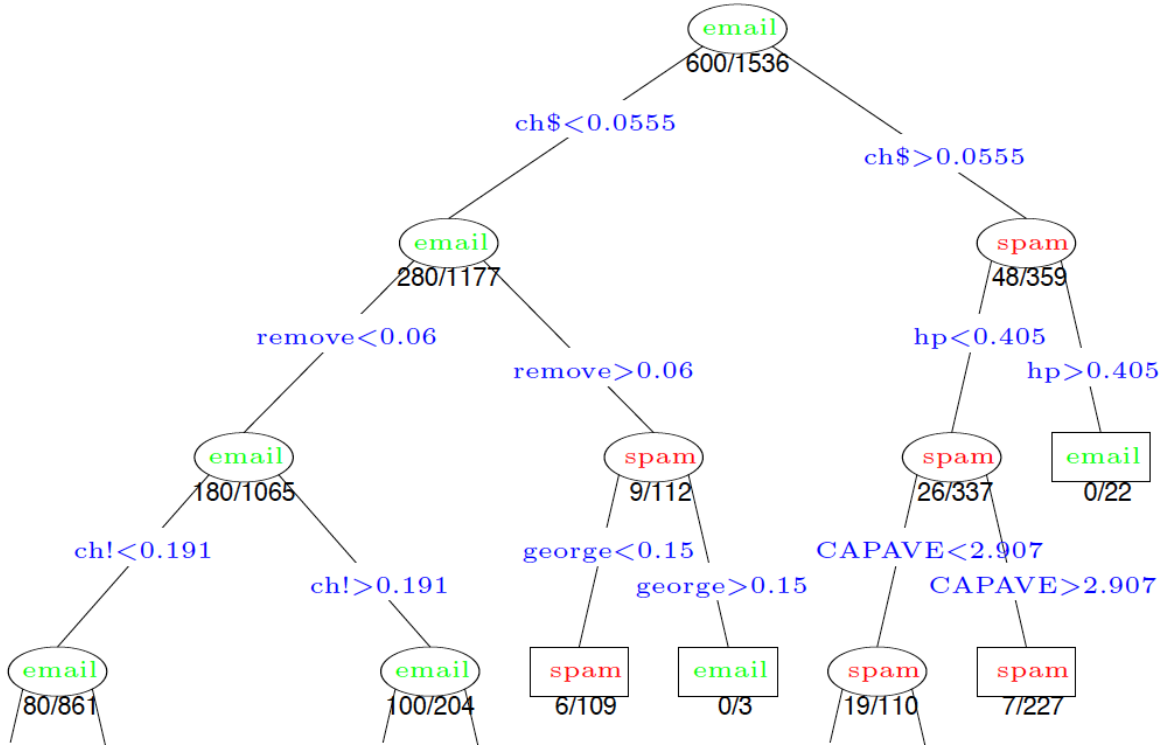
In this lecture, we learned more about logistic regression. We saw that is draws linear decision boundaries between the classes (linear in the predictor variable $x \in \mathbb{R}^p$), and learned how to interpret the coefficients, in terms of a multiplicative change in the odds ratio

We also compared logistic regression and LDA. Essentially, logistic regression is more robust because it doesn't assume normality, and LDA performs better if the normal assumption is (close to) true

We learned several model-free classification techniques. Typically these don't help with understanding the relationship between the predictors and the classes, but they can perform well in terms of prediction error. The k-nearest-neighbors method classifies a new input by looking at its k closest neighbors in the training set, and then classifying according to a majority vote among their labels. K-means clustering and learning vector quantization represent each class by a number of centroids or prototypes, and then use these to classify, and so they are more computationally efficient

Next time: tree-based methods

Classification trees are popular because they are easy to interpret



(From ESL page 315)