Lecture 5

IDS575: Statistical Models and Methods
Theja Tulabandhula

Notes derived from the book titled "Elements of Statistical Learning [2nd edition] (Sections 4.1-4.4, 7)

1 Classification using Linear Regression

Lets start thinking of classification now. Here the output variable G takes discrete values.

Say, hypothetically, you divide the input region into a collection of regions and label them according to the true classification. These decision boundaries can be jagged or nice (e.g., smooth/linear). There are methods that can output classifiers that only lead to linear decision boundaries. These are the ones we will start looking at.

As we saw before, we could use linear regression for classification. To do so, from G, make a K dimensional output variable Y (this is also called *one-hot encoding*). And fit a linear model. More precisely, predict the k^{th} indicator output variable using

$$\widehat{\mathrm{f}}_k(X) = \widehat{\beta}_k^T X.$$

The decision boundary between any two classes, say class k and l, would be given by $\widehat{\mathrm{f}}_k(X) = \widehat{\mathrm{f}}_l(X)$. This is a hyperplane[1]. Hence the input space is divided into regions with linear decision boundaries.

How does one classify using $\widehat{\mathrm{f}}_k, k = 1, ..., K$? Well, we can simply take the class corresponding to the largest one. These are generally called *discriminant functions*. We will see other discriminant functions later (LDA and logistic regression) soon.

Example 1. Linearity decision boundaries may seem more restrictive than necessary. But it turns out that certain nonlinear boundaries, such as quadratic, can be thought of as linear boundaries in a higher dimension. This is illustrated in Figure 1, where the original data is 2-dimensional.

Here are the steps for using linear regression for classification:

[1] Given constants a (p-dimensional) and c (1-dimensional), a hyperplane is a set of points z that obey $a^T z = c$.

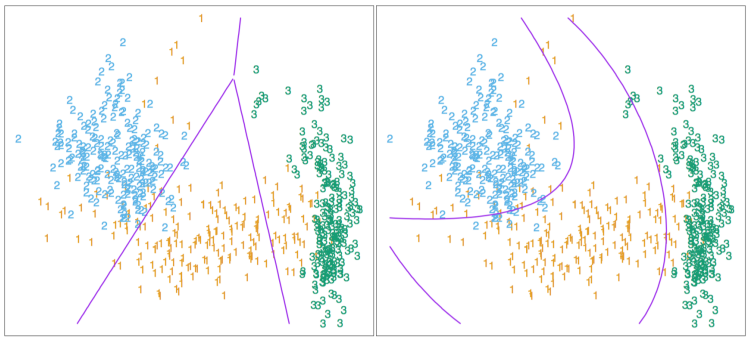

Figure 1: Linear decision boundaries in 2 and 5 dimensions (in the latter case, visualized in the original 2-dimensional space).

1. Each response category ($k = 1, ..., K$) is coded as an *indicator variable* (this is a K-dimensional vector with all zeros except the location/coordinate corresponding to category/level k.)
2. Collect them together to get the indicator response matrix $\mathbf{Y}$, which is $N \times K$-dimensional. Each row has a single 1 in it.
3. Fit a linear regression model to each of the columns of $\mathbf{Y}$ simultaneously, which is $\widehat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$. Here $\widehat{\beta}$ is $p \times K$-dimensional.
4. A new observation x_0 is classified by:
   - Compute $x_0^T \widehat{\beta}$, which is a K-dimensional row vector.
   - Identify the largest component, and report its index as the category/level.

Regression estimates conditional expectation. For Y_k, $E[Y_k|X = x] = Pr(G = k|X = x)$, hence this is reasonable, assuming $Pr(G = k|X = x)$ are linear functions of x.

Note 1. Some coordinates of $x_0^T \widehat{\beta}$ can be negative or greater than one, which is a bad approximation to $Pr(G = k|X = x)$.

Note 2. There is another issue with using regression, especially when $K \geq 3$: classes can get *masked* by other classes.

Example 2. Figure 2 represents an example of class masking. Here the three classes can be separated easily by linear boundaries, but linear regression is unable to do so! To understand this, look at Figure 3. The data has been projected onto the line joining the three centroids. The three regression lines are also plotted, and we can see that the line corresponding to the middle class is horizontal/never-dominant. Although using quadratic features help in this example, there is no easy solution in general.

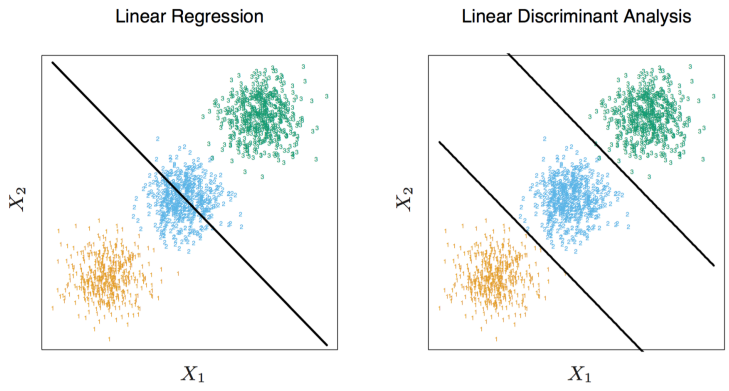

Figure 2: Class masking when linear regression is used for classification (left). Another method LDA (right) is able to circumvent this issue.

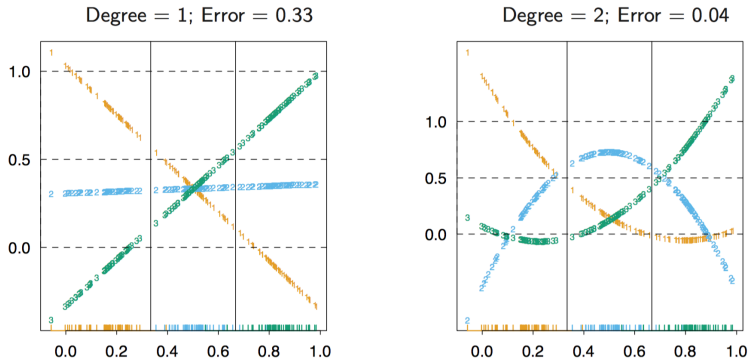

Figure 3: Class masking is minitgated using quadratic derived features (right) in this example. Without them, the $\widehat{\mathrm{f}}$ for one of the class never dominates (left).

2 Classification using Linear Discriminant Analysis (LDA)

If we recall statistical decision theory that we discussed before, then we know that $P(G = k|X = x)$ gives us the best classification (for the zero-one loss). Lets try to get to that quantity by using Bayes rule. Here's how:

1. Let the class conditional density of X be $f_k(X), k = 1, ..., K$
2. Let the class prior be $\pi_k, k = 1, ..., K$
3. Then the class posterior density is

$$P(G = k|X = x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l}.$$

The Bayes rule above tells us that choosing a model for $f_k(X)$ can give us the posterior distribution. There are many methods that model these densities, including LDA and *Naive Bayes* and Quadratic Discriminant Analysis (QDA). Lets focus on LDA for now.

LDA assumes that $f_k(X) \sim N(\mu_k, \Sigma_k)$ and $\Sigma_k = \Sigma$ for all $k = 1, .., K$.

Thus, when we compare two classes via the log-ratio of their posteriors we get:

$$\begin{aligned}\log \frac{Pr(G=k|X=x)}{Pr(G=l|X=x)} &= \log \frac{f_k(X)}{f_l(X)} + \log \frac{\pi_k}{\pi_l} \\ &= \log \frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k + \mu_l)^T \Sigma^{-1}(\mu_k - \mu_l) + x^T \Sigma^{-1}(\mu_k - \mu_l).\end{aligned}$$

This is linear in x!

The linear log-odds ratio implies that the decision boundary for any pair of *classes* is linear (the points x at which $Pr(G = k|X = x) = Pr(G = l|X = x)$).

Example 3. Figure 4 illustrates these boundaries for a simulated 2-dimensional data involving three classes.

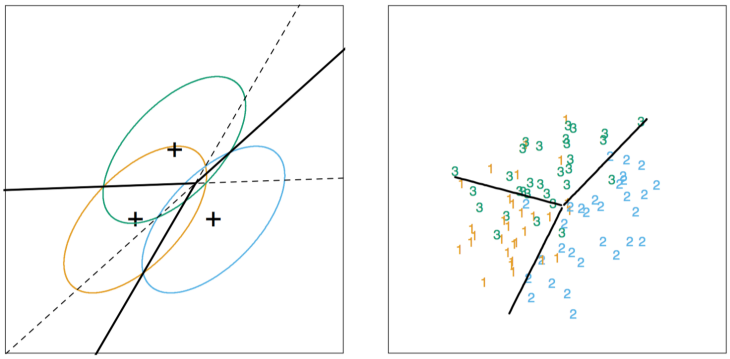

Figure 4: LDA boundaries for a simulated data with the same covariance for the three classes. Left plot shows the lines correspond to the Bayes decision boundaries. The right plot shows the LDA decision boundaries using training data.

The LDA decision rule is to pick the class with the maximum among the following *discriminant functions*: $\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$.

The parameters of the discriminant functions are estimated from training data as:

- $\widehat{\pi}_k = \frac{N_k}{N}$, where N_k is the number of class-k observations
- $\widehat{\mu}_k = \sum_{g_i=k} x_i / N_k$
- $\widehat{\Sigma} = \frac{1}{N-K} \sum_k \sum_{g_i=k} (x_i - \widehat{\mu}_k)(x_i - \widehat{\mu}_k)^T$.

Note 3. Gaussian assumption is critical here. Would it hold if one of the features was categorical?

Note 4. Quadratic Discriminant Analysis(QDA): If you relax the assumption that all class densities do not have the same Σ, we end up with discriminant functions that are quadratic. This also leads to quadratic decision boundaries.

Example 4. Figure 5 shows how QDA varies from LDA for a 2-dimensional dataset. For LDA, interactions terms were accounted for to fit a model in 5 dimensions (see Figure 1). As you can see, there is not much difference between them for this dataset. Since QDA needs to estimate different covariance matrices, the number of parameters is much larger.

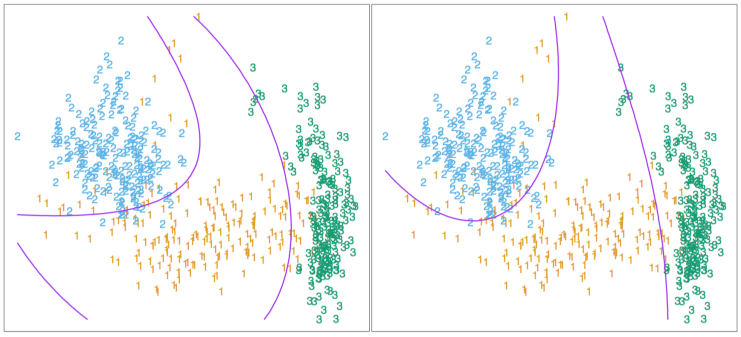

Figure 5: QDA (right) versus LDA (left).

3 Classification using Logistic Regression

Logistic regression model is defined using log-ratios of posterior probabilities of classes given feature vector. That is,

$$\log \frac{Pr(G = l|X = x)}{Pr(G = K|X = x)} = \beta_l^T x,$$

for $l = 1, ..., K-1$. The log ratio is also called a logit transformation, hence the name.

From these $K-1$ ratios, we can deduce that for each k

$$Pr(G = k|X = x) = \frac{\exp(\beta_k^T x)}{1 + \sum_{l=1}^{K-1} \exp(\beta_l^T x)}.$$

We will use maximum likelihood method to fit the model parameters. The log-likelihood given a training set of size N ($\{x_i, g_i\}_{i=1}^N$) is

$$l(\{\beta_1, ..., \beta_{K-1}\}) = \sum_{i=1}^{N} \log Pr(G = g_i|X = x_i)$$

Example 5. When $K = 2$, we can use variables y_i instead of g_i to write down the log-likelihood in a nicer form.

- Let $y_i = 1$ when $g_i = 1$ and let it be 0 when g_i is 2.
- Let $Pr(G = 1|X = x) = p$ (just a shorthand notation)

Then,

$$\begin{aligned} l(\beta_1) &= \sum_{i=1}^{N} \Big\{ y_i \log p + (1 - y_i) \log(1 - p) \Big\} \\ &= \sum_{i=1}^{N} \Big\{ y_i \beta^T x_i - \log(1 + \exp(\beta^T x)) \Big\}. \end{aligned}$$

Maximizing the log-likelihood turns out to be a convex[2] problem and can be solved using several off-the-shelf solvers.

Note 5. There is also another way to solve for β_1 that involves repeatedly solving linear regression problems. The method is called IRLS (Iteratively Reweighted Least Squares) that solves the following in each iteration:

$$\beta^{new} \leftarrow \arg\min_{\beta} (\mathbf{z} - \mathbf{X}\,\beta)^T (\mathbf{z} - \mathbf{X}\,\beta),$$

where $\mathbf{z} = \mathbf{X}\,\beta^{old} + \mathbf{W}^{-1}(\mathbf{y} - \mathbf{p})$, $\mathbf{W}$ is a $N \times N$ diagonal matrix with entries $Pr(G = 1|X = x_i)(1 - Pr(G = 1|X = x_i))$ and $\mathbf{p}$ is a N-dimensional vector with entries $Pr(G = 1|X = x_i)$, all evaluated at β^{old}. Finally, note that this is not super interesting unless one cares about optimization and is working in a large scale setting.

3.1 Logistic Regression versus LDA

Since the logistic model and the LDA have very similar forms for the log-ratios of probabilities, they may seem similar. They do in fact have the same linear form. But there are key differences in how the coefficients are estimated.

- Logistic regression makes lesser assumptions. Specifically, it does not impose that the marginal density of X is a mixture of Gaussian distributions.
- LDA on the other hand makes Gaussian assumptions. It is also estimated easily because of this.
- If indeed data was such that $P(X)$ was a mixture of Gaussians, then LDA would need lesser data to reach the same level of estimation accuracy.

So if you are unsure about which to use, side with logistic regression.

4 Model Selection and Assessment

Now that we have seen several methods: (a) linear regression with subset selection, ridge and LASSO penalties, (b) k-nearest neighbor methods, (c) linear discriminant analysis, and (d) logistic regression, lets discuss how to pick the most promising model.

We want to know the generalization performance: the prediction capability on independent test data.

This is the problem of *model selection and assessment*. We have already seen cross-validation as a way to assess any choice that we make while doing model search. There are other methods, some simpler than other that help us assess performance. Before doing that, lets understand *model complexity*.

4.1 Model Complexity, Model Bias and Model Variance

As before, let $\tau = \{(x_1, y_1), ..., (x_i, y_i), ..., (x_N, y_N)\}$ represent the training set, and $L(Y, \widehat{\mathrm{f}}_\alpha(X))$ denote the loss function for measuring errors (α denotes a tunable parameter or choice that you make). We have already seen the expected prediction error:

$$Err = E_{\tau, Pr(X,Y)}[L(Y, \widehat{\mathrm{f}}_\alpha(X))] = E_\tau[Err_\tau],$$

where $Err_\tau = E_{Pr(X,Y)}[L(Y, \widehat{\mathrm{f}}_\alpha)|\tau]$. This latter quantity is also what we want to know, but is not easy to estimate, so we focus on Err instead. Figure 6 depicts both of these for 100 simulated trainings sets with 50 observations each, and using LASSO.

Training error is the average loss over the training sample (RSS is a special case), and we will denote it as $\overline{err} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \widehat{\mathrm{f}}_\alpha(x_i))$. In general, negative log-likelihoods can be used to define the loss function as well.

The intuition for model complexity is this: as the model becomes more complex (more parameters), it will be able to adapt to more underlying structure in the data. The model bias will decrease, but the model variance will increase.

Note 6. Although we discuss quantitative outputs here, most of our discussion can also encompass the qualitative output setting easily.

Note 7. We want to set the tuning parameter α such that we get the minimum Err.

Note 8. There are two related but distinct goals: (a) model selection that involves estimating different models (corresponding to different αs) to pick the best one; and (b) model assessment that involves estimating the Err for a given model.

Example 6. Say we have lots of data. Then we can do the following:

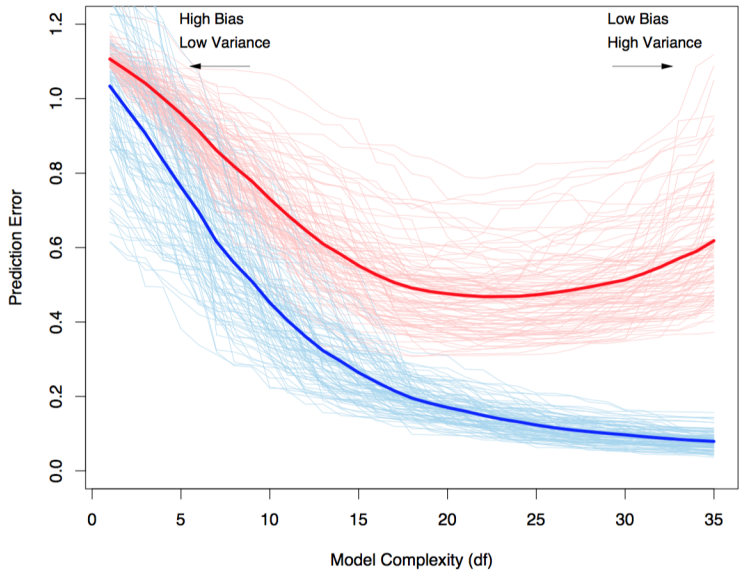

Figure 6: Illustration of expected prediction error Err (solid red) vs Err_τ (light red) for LASSO. Training error $\overline{err}$ (blue) decreases as model complexity increases.

- Randomly divide dataset into three parts as shown in Figure 7.
- Fit models using train.
- Use validation to estimate Err for model selection.
- Finally, use test set to estimate Err of the one chosen model (model assessment). Do not use test set multiple times!

Figure 7: Train, validation and test sets.

You have to make a choice about what proportion of your data will be validation and test.

If we do not have lots of data, then we can attempt to use other methods discussed here, viz., AIC, BIC and cross-validation.

4.2 Bias Variance Decomposition

Consider the data distribution $Pr(X, Y)$ such that $Y = f(X) + \epsilon, E[\epsilon] = 0, Var(\epsilon) = \sigma^2$, and X fixed. In this case, recall that the EPE (which is the same as Err for the squared

loss) can be written at a point x_0 as:

$$Err(x_0) = \sigma^2 + (E_\tau \widehat{\mathrm{f}}(x_0) - f(x_0))^2 + E_\tau(\widehat{\mathrm{f}}(x_0) - E_\tau \widehat{\mathrm{f}}(x_0))^2.$$

The first term is the variance of the output variable around its true mean $f(x_0)$. The second term is the squared model bias and the third term is model variance.

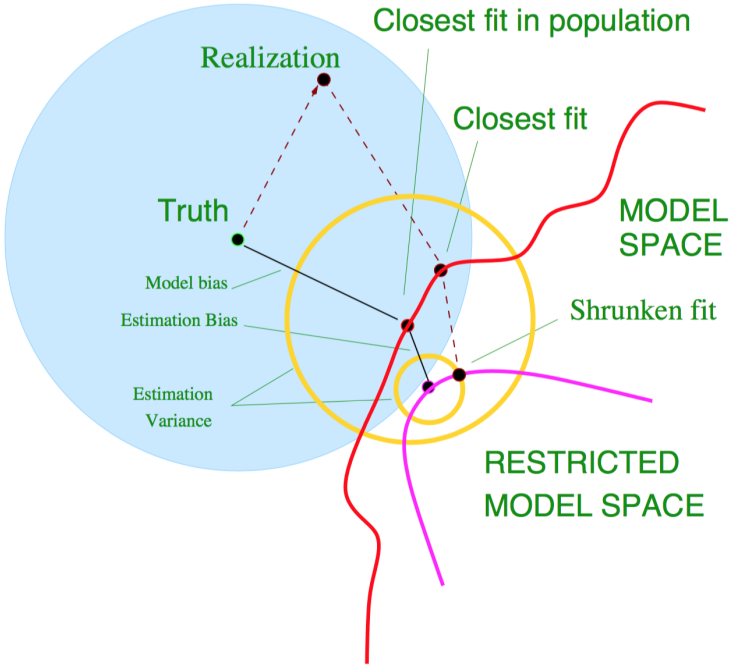

Figure 8: An example illustration of bias-variance decompostion. Blue region indicates σ^2. Red boundary represents linear models. Purple boundary represents ridge models. Yellow circles represent model variance (larger one for linear regression and smaller one for ridge).

The rule of thumb is that if a model is complex, then its bias will be low, but variance will be high.

Example 7. For k-nearest neighbor:

- Bias: $(f(x_0) - \frac{1}{k}\sum_{l=1}^{k} f(x_{(l)}))^2$.
- Variance: σ^2/k.

This means that if k is chosen to be high, the bias is high and the variance is low. When k is small, $\widehat{\mathrm{f}}(x)$ can adapt itself better to the underlying $f(x)$.

Example 8. For a p-dimensional linear model, the variance term for $Err(x_0)$ is equal to $\|\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}x_0\|_2\sigma^2$. There is a slightly intuitive version for the in-sample model variance term for $\frac{1}{N}\sum_i Err(x_i)$, which is $p\sigma^2/N$. Figure 8 shows an illustration of bias-variance decomposition for this class of models.

Note 9. Bias variance decomposition behaves differently for different loss functions. This implies different tuning parameters for different settings.

4.3 Model Selection Basics

Note 10. Intuitively, training error $\overline{err}$ will be less than the expected error Err because the same data is used to fit the model as well as assess its error.

Methods such as AIC and BIC try to estimate the optimism in training error $\overline{err}$ and add a corrective term to get an estimate of Err. Thus, they use training data only, and mostly make sense[3] for linear models. On the other hand, methods such as cross-validation directly estimate Err, but use 'validation' data.

4.4 AIC: Akaike Information Criterion

The AIC tells us to pick the model with the smallest AIC over the set of models considered.

AIC for regression models is defined as follows. Let $\overline{err}_\alpha$ and $d(\alpha)$ be the training error and number of parameters of model $\widehat{\mathrm{f}}_\alpha(x)$. Then, AIC is:

$$AIC = \overline{err}_\alpha + 2\frac{d(\alpha)}{N}\widehat{\sigma}^2,$$

where $\widehat{\sigma}^2$ is the estimated noise variance.

AIC for classification models such as logistic regression is defined as:

$$AIC = -\frac{2}{N}\sum_{i=1}^{N}\log Pr_{\widehat{\theta}}(y_i) + 2\frac{d}{N}.$$

Although it is a useful way to perform model selection, it may be inapplicable for certain choices of loss/likelihood functions. Further it is sensitive to $d(\alpha)$.

4.5 BIC: Bayesian Information Criterion

BIC is an alternative to AIC and is applicable when the loss function is defined in terms of a likelihood function (so we are maximum likelihoods here). The general formula is:

$$BIC = -2\sum_{i=1}^{N}\log Pr_{\widehat{\theta}}(y_i) + d\log N.$$

Under simple Gaussian assumptions, it simplifies to:

$$BIC = \frac{N}{\sigma^2}\overline{err} + d \log N.$$

Because of the $\log N$ term, it penalizes more complex models more heavily than AIC.

Although the formula looks very similar, BIC is motivated from a Bayesian perspective. Say we have multiple models. Say we have the same prior for each model. Then the posterior given data can be written as the likelihood times the prior. It turns out that the (negative of) log of this posterior can be approximated using the equations defined above.

Note 11. There is an additional benefit of BIC over AIC because of the Bayesian perspective: we can compare two models easily because we have the posterior probability of each model.

4.6 Nuances of Cross-Validation

Cross-validation (CV) estimates Err directly. It uses part of training data to fit models, and a different part to estimate Err, and repeats this by exchanging the parts. For example, $K = 5$ parts look as shown in Figure 9, where we have chosen the third part to estimate Err in this turn.

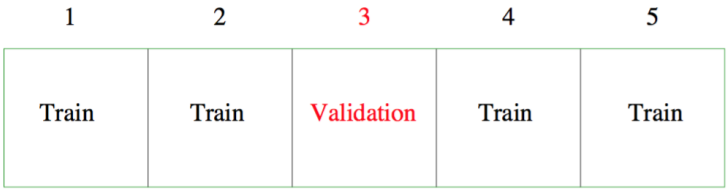

Figure 9: An example CV data split.

When $K = N$, we call this version of CV *leave-one-out* cross-validation.

How to pick K, the cross-validation parameter? Again it is a choice. Lets see how it impacts estimation of Err.

Example 9. One of the factors that influences how K-CV performs is the performance of the learning methods as training data increases. Figure 10 shows an illustration of the *learning curve* of a classifier. The classifier performance increases as training set size N increases up to about 100 observations, beyond which the performance improvement is small.

If our training data had 200 observations to work with. Then choosing $K = 5$ would give us 160 observations to fit a model, whose performance would be similar to the model fit using the full 200 observations. On the other hand, if our training data had 50 observations, then $K = 5$ would give us only 40 observations to fit a model, which would give us an incorrect (lower) estimate of $1 - Err$.

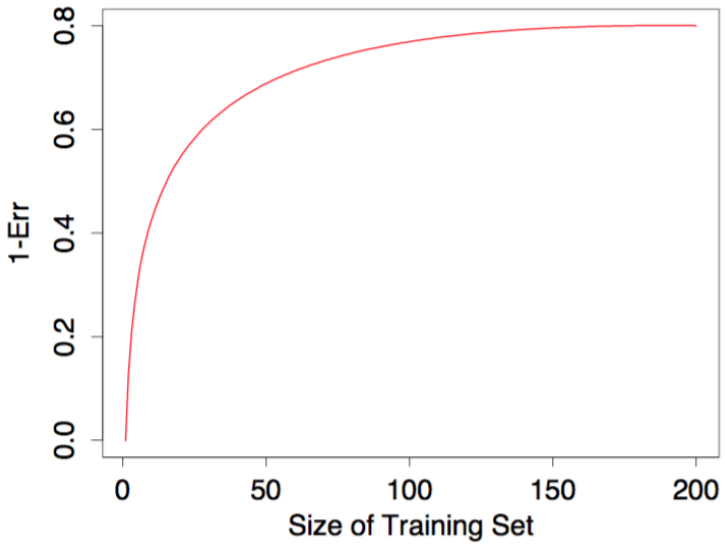

Figure 10: Learning curve as a function of training set.

4.7 Incorrect Use of CV

Cross validation should always be applied before the entire sequence (if any) of choices are made. For example, folds must be created before any filtering/data-processing steps are applied. And these filtering/data-processing steps must be applied in turn within each run of CV.

Note 12. If you ignore the realizations of the output variable, then you can process the remaining data (predictors) without involving cross-validation here.

Example 10. Say you plot scatterplots of predictors with the output and pick a subset of predictors (say 100). Using these, you build your classifier. And use cross-validation to estimate the error of the final model. *This is an incorrect application of cross-validation!*

Consider setting where $N = 50, p = 5000$ and all the input variables are standard Gaussians that are independent of class labels. Then the true $Err = 50\%$. If we use the above process for 1-nearest neighbor, we get a CV error rate as 3% (50 simulations, see Figure 11). The reason for this discrepancy is that the chosen input variables have an unfair advantage because they were chosen based on all data. Leaving data out after that does not let us estimate its Err correctly because these variables have "seen the left out data" as well. In Figure 11, we chose a random set of 10 observations (because we are using 5-fold CV) and computed the correlation between the pre-selected 100 input variable coordinates and the class labels of just these 10 samples. These correlations are non-zero!

So what is the correct way? First divide data into K folds. For each fold, find input coordinates that correlated with labels using all data except the k^{th} fold. Build a model

using all data except the k^{th} fold. Get its Err estimate. Repeat this over all folds in turn. Average the estimates. The bottom panel in Figure 11 shows the correlation of class labels with chosen predictors in a typical fold, which is zero on average.

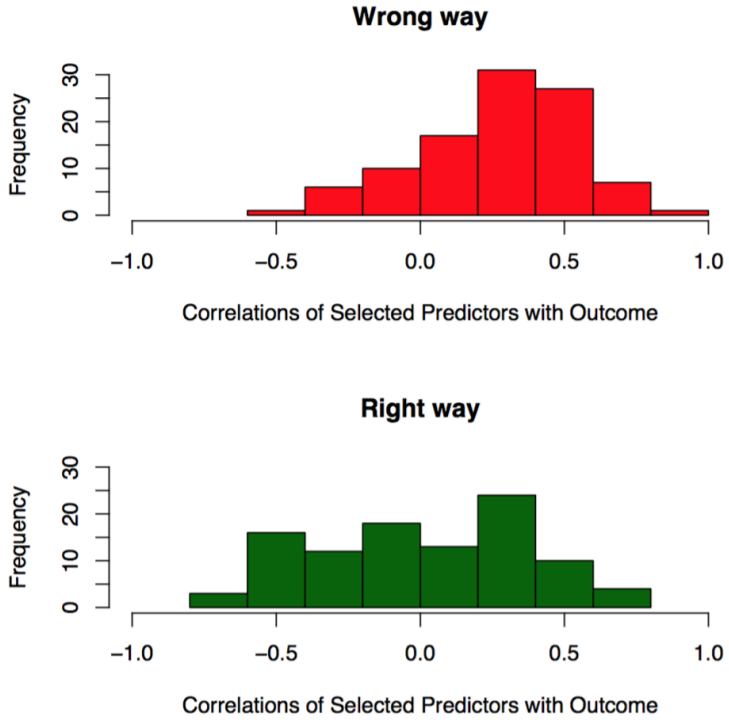

Figure 11: Incorrect vs correct use of cross-validation.

Note 13. Models must always be retrained from scratch for each turn of the K-CV.

Note 14. It is also useful to report the standard error of the CV estimate of Err to get an idea about the variance of the estimator.

5 Summary

We learned the following things:

- Classification via regression, linear discriminant analysis and logistic regression.
- Model selection and assessment using: (a) the bias-variance decomposition, (b) Akaike and Bayesian Information Criterion (AIC, BIC), and (c) cross-validation.

A Sample Exam Questions

1. What is class masking?
2. What are the similarities and differences between LDA and logistic regression?
3. What is the difference between Err and Err_τ?
4. How do you find the final model using cross-validation? What data will be used to find it?