Lecture 6: Methods for high-dimensional problems

Hector Corrada Bravo and Rafael A. Irizarry

March, 2010

In this Section we will discuss methods where data lies on high-dimensional spaces. In particular, we will be interested in problems where there are relatively few data points with which to estimate predictive functions. This is referred to as $N << p$ settings.

The curse of dimensionality

We have discussed the bias-variance tradeoff, and how flexible classifiers can, when model selection is properly applied, give us much better predictive performence. However, why would we ever consider a high-bias method such as linear regression? Is properly tuned KNN always better?

Consider the case where we have many covariates. We want to use kernel smoother. These methods can be generalized to cases where we have more than one or two predictors. Basically, we need to define distance and look for small multi-dimensional "balls" around the target points. With many covariate this becomes difficult. To understand why we need some mathematical intuition. Let's try our best to see why.

Imagine we have equally spaced data and that each covariate is in $[0, 1]$. We want to something like kNN with a local focus that uses 10% of the data in the local fitting. If we have p covariates and we are forming p-dimensional cubes, then each side of the cube must have size l determined by $l \times l \times \cdots \times l = l^p = .10$. If the number of covariates is p=10, then $l = .1^{1/10} = .8$. So it really isn't local! If we reduce the percent of data we consider to 1%, $l = 0.63$. Still not very local. If we keep reducing the size of the neighborhoods we will end up with very small number of data points in each average and thus with very large variance. This is known as *the curse of dimensionality*.

Because of this so-called curse, it is not always possible to use KNN and kernel smoothers. But other methods, like CART, thrive on multidimensional data.

SVD, PCA and ridge regression

First, let's revisit the geometry of linear regression as this will help us understand some of the issues in high-dimensional problems.

Consider data matrix $\mathbf{X}$. A useful tool to visualize and predict is to look at the *principal components* of the variables in $\mathbf{X}$.

Assume the $\mathbf{X}$ are centered, i.e. $\mathbf{X}\underline{1} = \underline{0}$, the sample covariance matrix is given by $\mathbf{X}'\mathbf{X}/N$ and $\mathbf{X}'\mathbf{X}$ can be written as

$$\mathbf{X}'\mathbf{X} = \mathbf{V}\mathbf{D}^2\mathbf{V}.$$

Technical note: this is the eigen decomposition of $\mathbf{X}'\mathbf{X}$.

The v_js are called the eigen-vectors of the sample covariance matrix and also the principal component directions of $\mathbf{X}$. Figure 1 shows a scatterplot of $\mathbf{X}$ and the directions as red (solid) and blue (dashed) lines.

The first principal component $\mathbf{z}_1 = \mathbf{X}v_1$ has the property that it has the largest sample covariance among all normalized (coefficients squared add up to 1) linear combinations of $\mathbf{X}$. The sample variance is d_1^2/N.

The derived variable $\mathbf{z}_1 = \mathbf{X}v_1 = \mathbf{u}_1 d_1$ is called the first principal component. Similarly $\mathbf{z}_j = \mathbf{X}v_1$ is called the jth principal component. $\mathbf{XV} = \mathbf{UD}$ is a matrix with principal components in the columns. Figure 2 shows these.

A related decomposition is the singular value decomposition (SVD) of the centered input matrix $\mathbf{X}$. This decomposition is extremely useful in many statistical analysis methods. We will see it again later.

The SVD of an $N \times p$ matrix $\mathbf{X}$ is

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}'$$

with $\mathbf{U}$ and $\mathbf{V}$ $N \times p$ and $p \times p$ orthogonal matrices and $\mathbf{D}$ a $p \times p$ diagonal matrix with entries $d_1 \geq d_2 \geq \ldots d_p \geq 0$ called the singular values of $\mathbf{X}$.

Technical Note: $\mathbf{U}$ is an orthogonal basis for the space defined by the columns of $\mathbf{X}$ and $\mathbf{V}$ is an orthogonal basis for the space defined by the rows of $\mathbf{X}$.

We can show that the least squares predictor for linear regression is

$$\begin{aligned} \hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}}^{ls} &= \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \\ &= \mathbf{U}\mathbf{U}'\mathbf{y} \end{aligned}$$

Technical Note: $\mathbf{U}'\mathbf{y}$ are the coordinates of $\mathbf{y}$ with respect to the orthogonal basis $\mathbf{U}$

Recall the ridge regression problem, written in *Lagrangian* form:

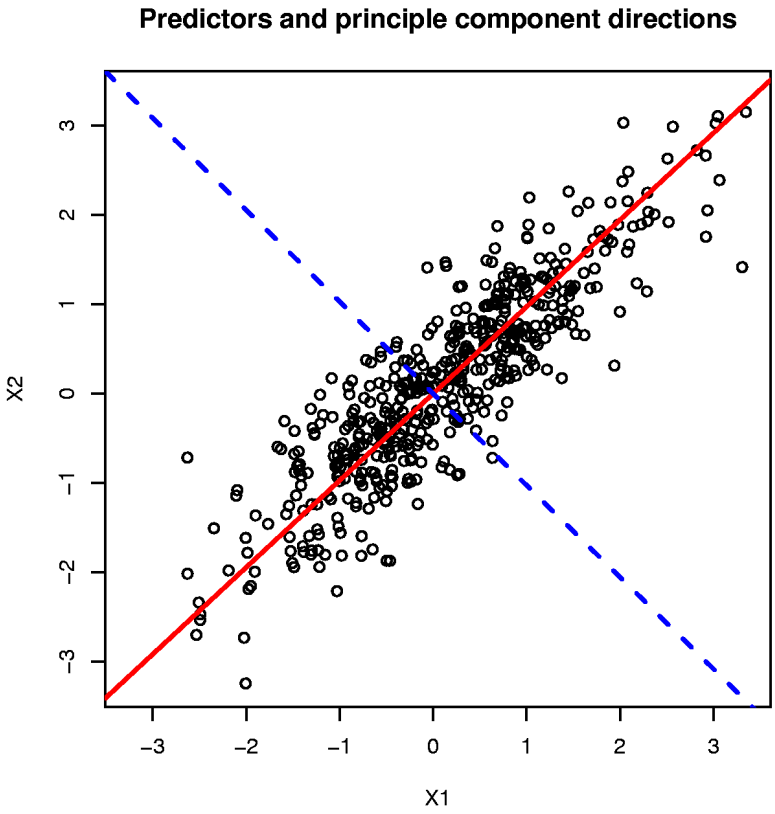

Figure 1: Plot of two predictors, X_2 versus X_1, and the principal component directions

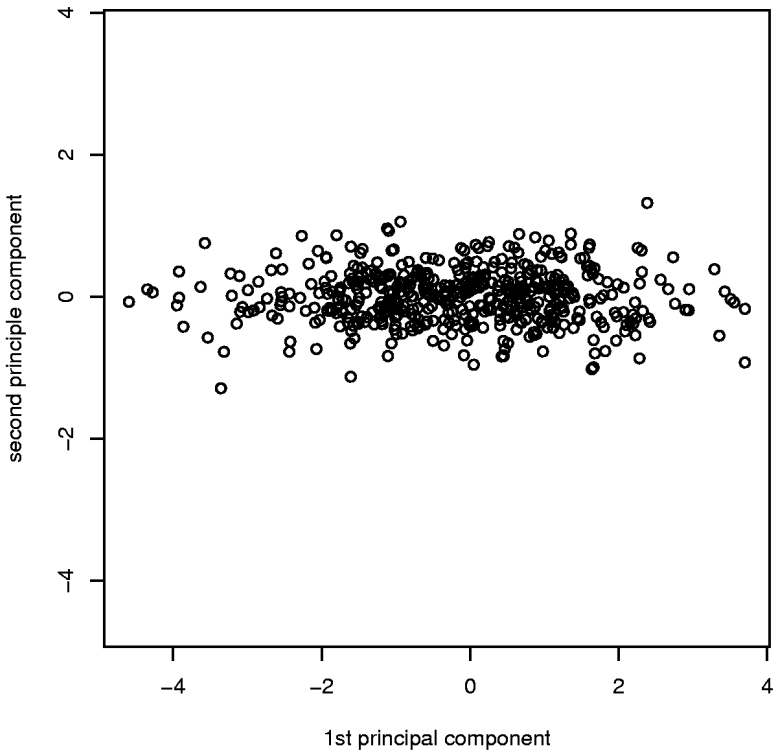

Figure 2: Plot of two principal components of $\mathbf{X}$.

$$\min_{\beta} \sum_{i=1}^{N} (y_i - \beta_0 - \beta' x_i) + \lambda \sum_{j=1}^{p} \beta_j^2$$

.

The ridge solution can be expressed as

$$\begin{aligned} \mathbf{X}\hat{\boldsymbol{\beta}}^{\text{ridge}} &= \mathbf{X}(\mathbf{X}'\mathbf{X} + \lambda \mathbf{I})^{-1}\mathbf{X}'y \\ &= \mathbf{U}\mathbf{D}(\mathbf{D}^2 + \lambda I)^{-1}\mathbf{D}\mathbf{U}'y \\ &= \sum_{j=1}^{p} \mathbf{u}_j \frac{d_j}{d_j + \lambda} \mathbf{u}_j' \mathbf{y} \end{aligned}$$

Notice that because $\lambda > 0$, $\frac{d_j}{d_j+\lambda} \leq 1$. Like linear regression, ridge regression computes the coordinates of $\mathbf{y}$ with respect to the orthogonal basis $\mathbf{U}$. It then shrinks these coordinates by the factors $\frac{d_j}{d_j+\lambda}$. This means that a greater amount of shrinkage occurs when λ is big and for smaller d_js.

We now see that ridge regression shrinks coefficients related to principal components with small variance. This makes sense because we have less information about them.

In the case of Figure 1, we can think of it as weight and height, we are saying predict with the sum and ignore the difference. In this case, the sum give much more info than the difference.

Ridge and Lasso comparison

Suppose data matrix $\mathbf{X}$ is orthonormal, i.e. the predictors are not correlated and in, loosely, in the same scale. Then from the above we can see what best subset regression, ridge regression and the lasso do to the least-squares estimate $\hat{\beta}_j$ (recall that in this case the least squares estimate are "decoupled" and are simple projections of the outcome onto each predictor).

Ridge regression *shrinks* the estimate as we saw above: $\hat{\beta}_j^{\text{ridge}} = \frac{\hat{\beta}_j}{1+\lambda}$. The best subset (of size M) includes predictor j if it's least squares estimate $\hat{\beta}_j$ is one of the M largest in absolute value. That is, the best subset estimate is $\hat{\beta}_j \cdot I\{\text{rank}(|\hat{\beta}_j|) \leq M\}$.

Recall the lasso in *Lagrangian* form:

$$\min_{\beta} \sum_{i=1}^{N} \left(y_i - \beta_0 - x_i'\beta\right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

The lasso does a soft version of best subset: $\hat{\beta}^{\text{lasso}} = \text{sign}(\hat{\beta}_j)(\hat{\beta}_j - \lambda)_+$, where $(x)_+$ is the *positive part* of x. Whereas best subset does *hard thresholding*, the

lasso does *soft thresholding*. Figure 3 illustrates these transformations on the least square estimate.

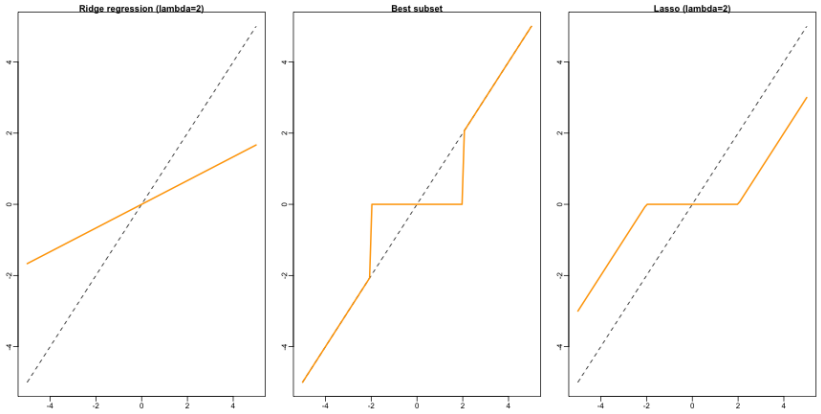

Figure 3: Transformations of the least squares estimate

LDA

Let's recall the computations required for LDA and QDA. Suppose we compute the eigen-decomposition of each $\hat{\mathbf{\Sigma}}_k = \mathbf{U}_k \mathbf{D}_k^2 \mathbf{U}_k'$, where $\mathbf{U}_k$ is a $p \times p$ matrix with orthonormal columns and $\mathbf{D}_k^2$ is a diagonal matrix of positive eigenvalues d_{kl}. The ingredients for $\delta_k(x)$ are:

- $(x - \hat{\mu}_k)' \mathbf{\Sigma}_k^{-1} (x - \hat{\mu}_k) = [\mathbf{U}_k'(x - \hat{\mu}_k)'] \mathbf{D}_k^{-2} [(x - \hat{\mu}_k)\mathbf{U}_k]$
- $\log |\mathbf{\Sigma}_k|^{-1} = \sum_l \log d_{kl}$

Notice this is much easier to compute since $\mathbf{D}_k^2$ is a diagonal matrix!

Given this, we can now compute and interpret the LDA classifier as follows:

- *Sphere* the data with respect to the common covariance estimate $\hat{\mathbf{\Sigma}}$ to get $X^* = \mathbf{D}\mathbf{U}'X$. The common covariance estimate of X^* is now the identity matrix!
- Classify to the closest class centroid (μ_ks) in the transformed space, correcting for the effect of the class prior probabilities π_k.

Section 4.3.3 of Hastie, Tibshirani and Friedman, has a nice discussion of how LDA is related to the solution of the problem: *find the linear combination $Z = a'X$ such that the between-class variance is maximized relative to the within-class variance.* How is this related to the first principal component?

Diagonal LDA and shrunken centroids

In high-dimensional problems, we might not have enough data to compute all parameters of the covariance matrix needed for LDA. A nice, but powerful, form of regularization is to estimate the covariance matrix assuming that predictors are independent, i.e., the covariance matrix is *diagonal*. How can we interpret this in terms of principal components?

The *diagonal-covariance LDA* discriminant score function for class k is

$$\delta_k(x^*) = -\sum_{j=1}^{p} \frac{(x_j^* - \bar{x}_{kj})^2}{s_j^2} + 2\log \pi_k.$$

Here $x^* = (x_1^*, x_2^*, \ldots, x_p^*)'$ is a vector corresponding to a new observation, s_j is the pooled within-class estimate of the standard-deviation of predictor j, and $\bar{x}_{kj} = \sum_{g_i=k} x_{ij}/N_k$ is the mean of the N_k values for predictor j in class k. We call $\bar{x}_k = (\bar{x}_{k1}, \bar{x}_{k2}, \ldots, \bar{x}_{kp})'$ the *centroid* of class k. The discriminant score is then the standardized distance between observation x^* and class centroid $\bar{x}_k$ adjusted by the class prior probability π_k.

The classification rule is then

$$\hat{G}(x^*) = \arg \max_{k=1,\ldots,K} \delta_k(x^*).$$

The diagonal LDA classifier is equivalent to *nearest centroid* classifier after appropriate standarization (and shift from the class priors).

To help with interpretation, we want to reduce the number of predictors used in the classifier. For instance, diagonal LDA uses all predictors when computing the standarized distance to each centroid. Preferrably, we want to remove predictors that do not contribute to the class predictions. Intuitively, these would be predictors for which the class means (or centroids) are close to each other, so that points are equi-distant to each of the class centroids.

One way of filtering predictors in this case is to use the two-sample t-statistic for each predictor:

$$t_j = \frac{\bar{x}_{2j} - \bar{x}_{1j}}{s_j}$$

The t_j-statistic gives a measure of how *significant* is the difference in the class means for predictor j. If the t_j values were normally distributed, then one may consider values with $|t_j| > 2$ significant. Figure4 shows predictors for a gene expression dataset comparing lung cancer vs. normal lung tissues. They are ordered by the t-statistic from left to right.

Similarily, consider the following statistic:

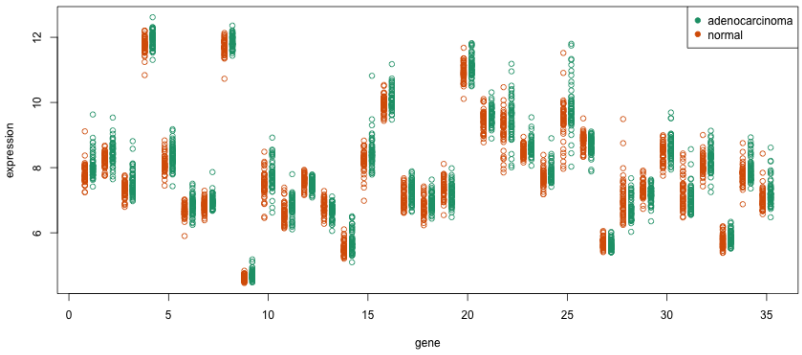

Figure 4: Top ranking predictors by t-statistic in a gene expression dataset

$$d_{kj} = \frac{\bar{x}_{kj} - \bar{x}_j}{m_k s_j},$$

with $m_k = 1/N_k - 1/N$. Notice that with constant within-class variance σ^2, the variance of the contrast $\bar{x}_{kj} - \bar{x}_j$ is $m_k^2\sigma^2$. This gives a measure of how *significant* is the difference between the class k mean for predictor j, and the overall mean for predictor j. In the two-class setting, how are the two statistics related?

The *nearest shrunken centroid* method uses a version of this statistic to *regularize* the nearest centroid predictor by *shrinking* the class means towards the overall mean for each predictor. Let

$$d_{kj} = \frac{\bar{x}_{kj} - \bar{x}_j}{m_k(s_j + s_0)},$$

with s_0 a small positive constant, typically the median of the s_j values, used to guard against d_{kj} values resulting from predictors with values near zero (this is common in gene expression data). Suppose you threshold these values:

$$d'_{kj} = d_{kj} \cdot I\{|d_{kj}| \geq \Delta\}.$$

Then $d'_{kj} = d_{kj}$ if the standarized class mean is "significantly" different from the overall mean, and zero otherwise. Now let's *shrink* the class means $\bar{x}_{kj}$ towards the overall mean as

$$\bar{x}_{kj} = \bar{x}_j + m_k(s_j + s_0)d'_{kj}.$$

What happens to class centroids with no *significant* difference with the overall centroid? Thus, unless a predictor has a *significant* difference to the overall mean for at least one class, it is useless for classification. How is this related to filtering predictors using the t-statistic?

Now, the thresholding we used is called *hard thresholding*, and it has some problems. Besides, selection, we want to use *shrinkage* to pool data when estimating p-dimensional centroids. Therefore, shrinking towards the overall mean (is estimated with all N points) is a good idea. So to do both selection and to pool data to get estimates we can use *soft thresholding*.

$$d'kj = \text{sign}(d_{kj})(|d_{kj} - \Delta|)_+,$$

The class centroids are shrunken towards the overall centroid as before. Δ is a parameter to be chosen (cross-validatated prediction error can be used). Recall how the lasso does *soft thresholding* of the least squares estimate in regression.

Quadratic regularization for linear classifiers

Regularized Discriminant Analysis

Suppose we want to use LDA but not require that the variance is diagonal. If $p >> N$ then the non-diagonal p-by$p$ estimate $\hat{\mathbf{\Sigma}}$ is of rank K and therefore $\hat{\mathbf{\Sigma}}^{-1}$ is undefined. That means we can't compute the discriminant direction $\hat{\mathbf{\Sigma}}^{-1}(\hat{\mu}_2 - \hat{\mu}_1)$. The *regularization* approach in this case is to *shrink* $\hat{\mathbf{\Sigma}}$ towards its diagonal:

$$\hat{\mathbf{\Sigma}}(\gamma) = \gamma\hat{\mathbf{\Sigma}} + (1-\gamma)\text{diag}(\hat{\mathbf{\Sigma}}),$$

with γ a regularization parameter to be chosen. This form of regularization is similar to what ridge regression does, can you see why? This can be combined with shrunken centroids, but distances are now "warped".

Regularized logistic regression

Consider the multinomial logistic regression model

$$\Pr(G = k|X = x) = \frac{\exp\{\beta_{0k} + x'\beta_k\}}{\sum_{l=1}^{K}\exp\{\beta_{0l} + x'\beta_l\}}$$

which has K coefficient vectors of log-odds parameters. We regularize by maximizing the *penalized* log-likelihood

$$\max_{\{\beta_{0k},\beta_k\}_1^K} \left[\sum_{i=1}^{N} \log Pr(g_i|x_i) - \lambda \sum_{k=1}^{K} \sum_{j=1}^{p} \beta_{kj}^2 \right].$$

Like ridge regression, the intercepts are not penalized.

Computational considerations for quadratic penalties

Recall the SVD of data matrix $\mathbf{X}$:

$$X = \mathbf{UDV}' \tag{1}$$
$$= \mathbf{RV}' \tag{2}$$

Remember what $\mathbf{R}$ is? Recall the ridge estimate:

$$\hat{\beta}^{\text{ridge}} = (\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}'\mathbf{y} \tag{3}$$
$$= \mathbf{V}(\mathbf{R}'\mathbf{R} + \lambda\mathbf{I})^{-1}\mathbf{R}'\mathbf{y} \tag{4}$$

Thus $\hat{\beta}^{\text{ridge}} = \mathbf{V}\hat{\theta}$, where $\hat{\theta}$ is the ridge estimate of the N observations (r_i, y_i), with r_i the ith row of matrix $\mathbf{R}$. Therefore, we can get the ridge estimate by reduce the data matrix from $\mathbf{X}$ to $\mathbf{R}$ and work with this smaller matrix. This is particularly handy when using cross-validation since you can use $\mathbf{R}$ for all values of regularization parameter λ.

L1 regularization for linear classifiers

Similarly to the above, we can penalize the multinomial logistic regression model with a lasso penalty:

$$\max_{\{\beta_{0k},\beta_k\}_1^K} \left[\sum_{i=1}^{N} \log Pr(g_i|x_i) - \lambda \sum_{k=1}^{K} \sum_{j=1}^{p} |\beta_{kj}| \right].$$

The `glmnet` package provides very fast code to fit these types of models.