MLISP: Machine Learning in Signal Processing

Lecture 3

Prof. V. I. Morgenshtern

Scribe: M. Solomon

Illustrations: The elements of statistical learning, Hastie, Tibshirani, Friedman

Agenda:

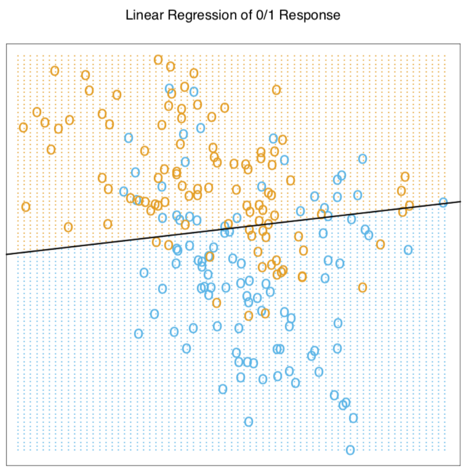
1. Linear model for classification
2. Nearest neighbors
3. Degrees of freedom, bias, variance

1 Linear model for classification

In the previous lecture, we have considered the example of regression problem, where the output variable (the house price) was continuous.

In classification problem, the output variable is *discrete*. For example, in digit classification, there are 10 possible discrete outcomes. Discrete variables are sometimes called *categorical*.

In the following figure, we see a scatter plot of training data for a pair of inputs x_1, x_2. The data is simulated, the details of the simulation are not important for now.



We can encode this data by using $y = 0$ if o and $y = 1$ if o, and then we can use the linear model similarly to what we did previously. Concretely, define the cost function as before:

$$J(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{n}\left(\langle \mathbf{x}^{(i)}, \boldsymbol{\theta}\rangle - y^{(i)}\right)^2. \tag{1}$$

The new element here is that the fitted values $\hat{y}$ are converted to a fitted class variable $\hat{g}$ according to the rule:

$$\hat{g} = \begin{cases} \text{o}, & \text{if } \hat{y} \le 0.5 \\ \text{o}, & \text{if } \hat{y} > 0.5. \end{cases}$$

This gives us the decision boundary: $\langle \mathbf{x}^{(i)}, \hat{\boldsymbol{\theta}}\rangle = 0.5$ which is linear. We see that for this data, there are several misclassification on both sides of the decision boundary.

Is our linear model too rigid or are such error unavoidable?

Remember that these errors are on the training data itself, and we have not said which distribution the simulated data came from. For example, let's consider the following two scenarios.

Scenario 1:
The training data in each class were generated as bi-variate Gaussian distribution with different means.

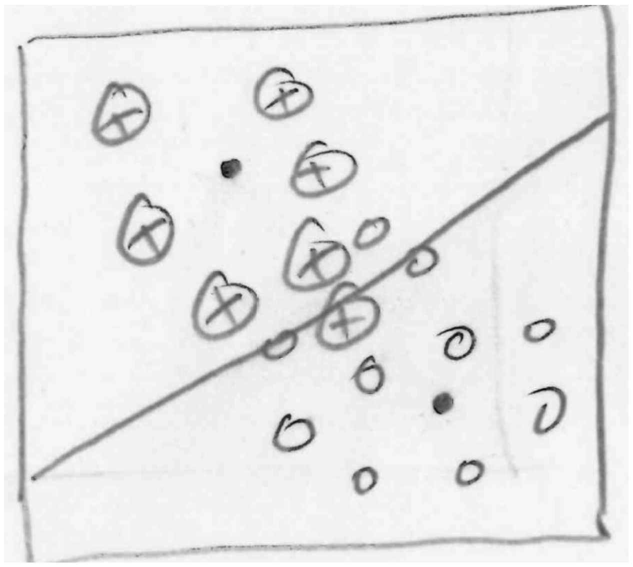


Figure 1: Simulated data with simple decision boundary.

Intuitively, linear decision boundary is optimal, overlap is inevitable. More details on this later.

Scenario 2:
The training data in each class came from a mixture of 10 low-variance Gaussians, with individual means themselves distributed as Gaussians.

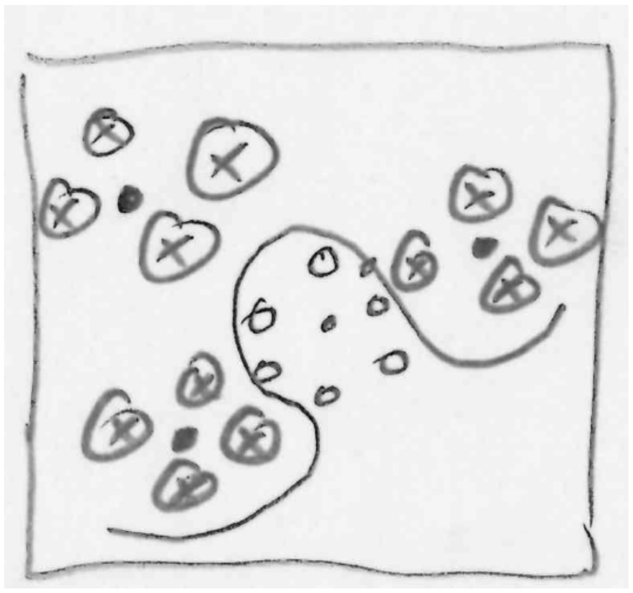


Figure 2: Simulated data with complex decision boundary.

Intuitively, the linear decision boundary is unlikely to be optimal, and in fact is not. The optimal decision boundary is non-linear and disjoint. Therefore it is much more difficult to obtain.

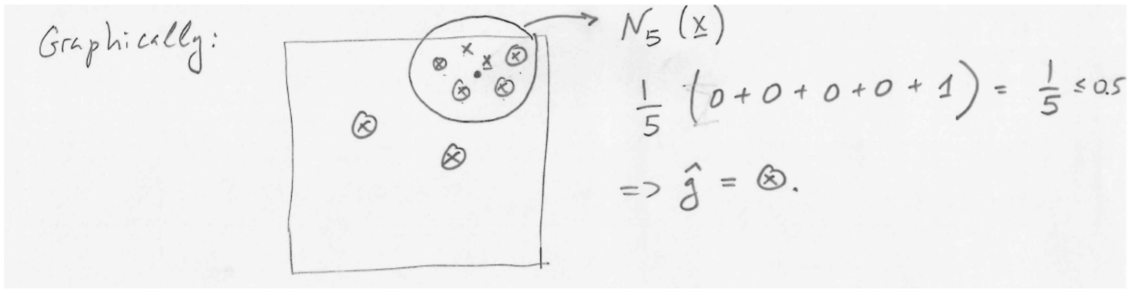
2 k-nearest neighbors algorithm

The k-nearest neighbor algorithm is much more flexible than the linear model and can adapt to the distributions as the one in Figure 2.

For a new data point $\mathbf{x}$, the prediction is:

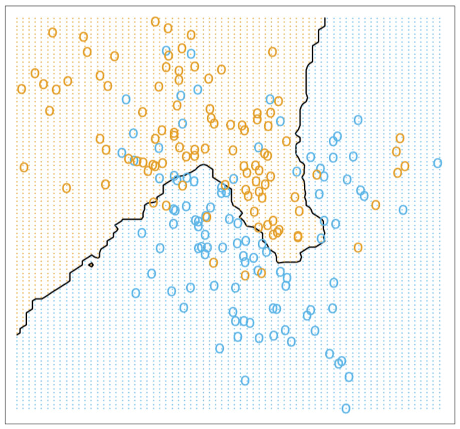
$$\hat{y}(\mathbf{x}) = \frac{1}{k} \sum_{\mathbf{x}^{(i)} \in N_k(\mathbf{x})} y^{(i)} \tag{2}$$

where $N_k(\mathbf{x})$ is the neighborhood of $\mathbf{x}$ defined by the k closest points $\mathbf{x}_i$ in the training sample.


Graphically:
$N_5(\underline{x})$
$\frac{1}{5}(0+0+0+0+1) = \frac{1}{5} \le 0.5$
$\Rightarrow \hat{y} = \otimes$.

Here is the result of 15-nearest-neighbor averaging of the binary coded response as the method of fitting.

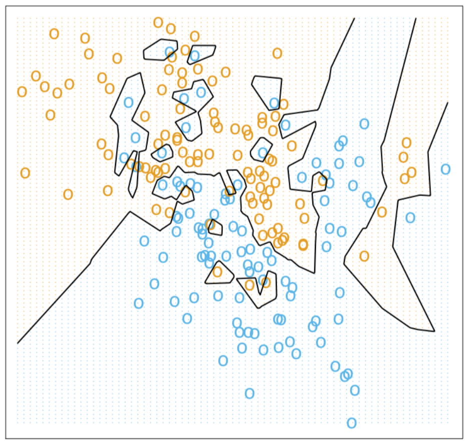
15-Nearest Neighbor Classifier



This is a much better fit than linear model.

For the 1-nearest-neighbor algorithm, we got a decision boundary like this:

1−Nearest Neighbor Classifier



It is clear that for the k-nearest neighbor fit, the error on the training set should approximately increase as a function of k, and it will always be 0 for $k = 1$.

Important: it is very easy to fit the training data perfectly. For example, the nearest neighbors algorithm with $k = 1$ accomplishes this task. However, fitting the training data perfectly does not guarantee that we will perform well on future data. We must have an independent test set to compare different methods!

3 Degrees of freedom, bias, variance

The number of Degrees of Freedom (DoF) of a statistical model is the number of independent parameters that can be adjusted.

For the linear model: $y = \theta_0 + \theta_1 x_1 + \theta_2 x_2$, the number of DoF is $p = 3$.

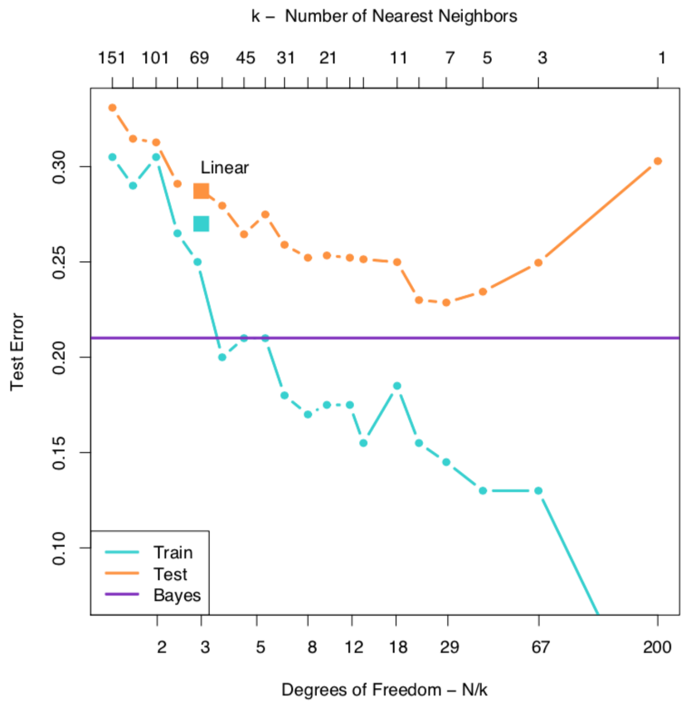
For the k-nearest neighbors the effective number of DoF is $\frac{n}{k}$, where n is the number of data points. To understand this intuitively, consider the simplified situation in which the neighborhoods do not overlap. Then, there are $\frac{n}{k}$ neighborhoods with k points each. For each neighborhood we learn the label 0 or 1, by the majority voting in that neighborhood. Hence, there are $\frac{n}{k}$ numbers to learn.

Let's compare the linear model and the k-nearest neighbors algorithm:

- Linear decision boundary produced by the linear model is very smooth, heavily relies on the assumption that linear decision boundary is appropriate, stable to fit. The model is not sensitive to the small perturbations of the training data; it has *low variance*. If the linear decision boundary is not appropriate, the model will make many mistakes even on the training set; it has *high bias*.
- k-nearest neighbor procedure does not rely on any stringent assumption about the underlying data. It produces a wiggly decision boundary which is sensitive to the perturbations of the training data; it has *high variance*. Since the model can automatically adapt to arbitrarily complex decision boundaries, it has *low bias*.

We will discuss the trade-off between the bias and the variance in more details below.

Here are the results of classifying 10000 new observations from the same model:



In the figure above, k is displayed on the y axis at the top, and the DoFs are displayed on the y axis at the bottom.

For the nearest neighbors algorithm, we can see that as the number of degrees of freedom grows, the error on the training set decreases. This is not true for the error computed on the test set. For

k smaller than about 9 the error on the test set begins to increase. The situation when the error on the test set decreases and the error on the train set increases is a typical sign indicating that the model over-fits. In practice, we would look at the graph like this and set $k = 9$ as an optimal choice for the problem. If we want to make k smaller, we need more data.

The performance of the linear model is depicted. We observe that the linear model behaves similarly to the nearest neighbors algorithm with large k ($k \approx 60$).