# Chapter 3

# Overview of Supervised Learning

## 3.1 One predictor examples

Most of what we talk about in this class can be called Supervised Learning. The task it to predict outcomes given predictors but we have at our disposal some preliminary data (which we will refer to training data). Today we give an overview. In later lectures we will describe specific methods that have a good reputation of working well.

Real examples: Fraud detection, heart attack prediction given medical history and vitals, tumor classification with microarrays.

What is considered random?

Illustrative (unrealistic example):

Does it look normal?

This is actually a histogram of US adult heights. For each person we record gender and height. If we observed height, how do we predict gender?

Here is a example of a sample of size 20:

```
67 69 67 65 66 63 62 68 62 61 70 72 69 69 63 74 71 66 64 68
```

If we have a *training set* can we use regression?

What kind of values will E$[Y|X=x]$ take?

Here is the training data:

```
X: 63 65 62 69 65 62 66 67 66 64 68 67 67 66 68 68 69 71 69 74
Y: 1  1  1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0
```

The data looks like this:

The regression function is $Y = 8.1 - 0.11X$ and the predictors are:

Figure 3.1: Histogram of outcomes of predictor

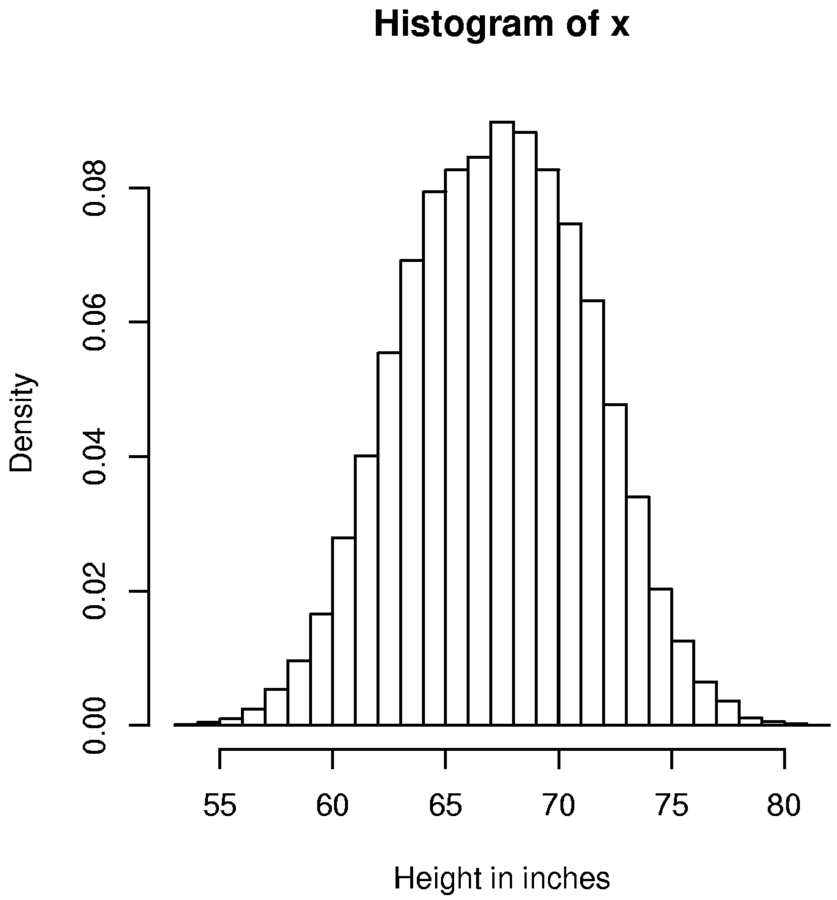


Figure 3.2: Histogram of outcome of predictors with both populations shown.

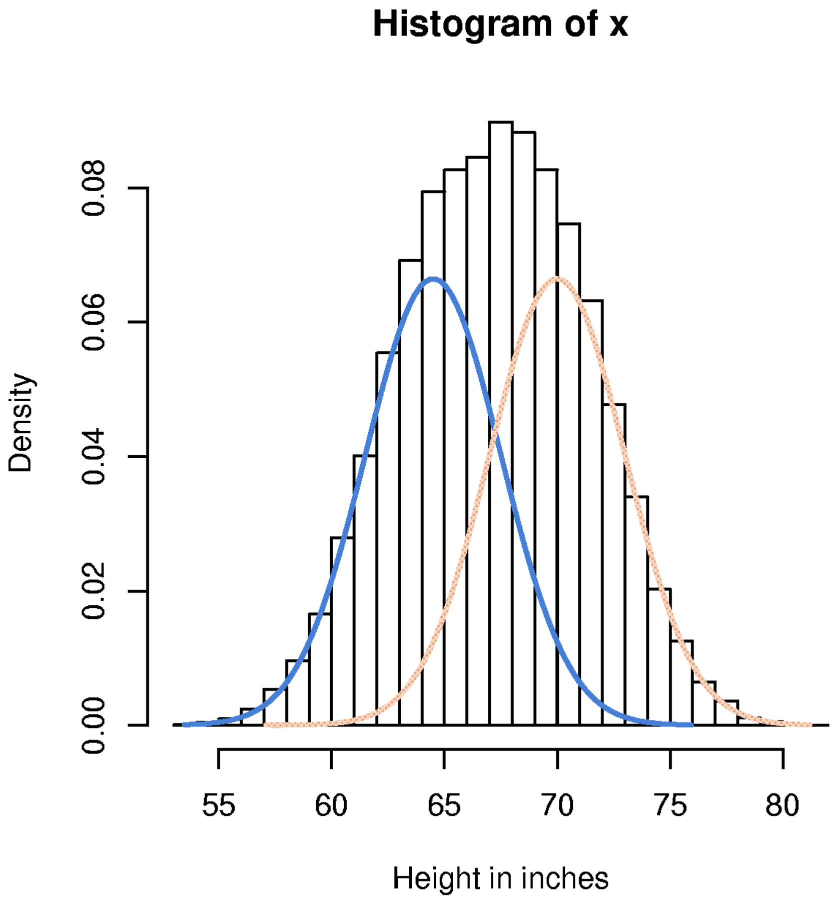
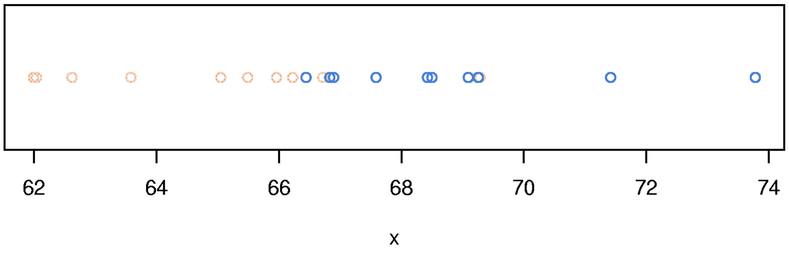


Figure 3.3: Histogram of predictor outcomes with two populations shown.



```
X:     63    65    62    69    65    62    66    67    66    64
Yhat: 0.98  0.71  1.05  0.22  0.66  1.05  0.60  0.52  0.57  0.87

X:     68    67    67    66    68    68    69    71    69    74
Yhat: 0.42  0.50  0.50  0.55  0.31  0.32  0.25 -0.02  0.23 -0.28
```

Seem like $\hat{Y} > 0.5$ does pretty well. Notice this translates to $X < 69$.

Turns out the best solution is $X < 67.25$. Why did not we get it?

Will the best solution be perfect? Is it better than guessing? Notice that if we pick a woman, see her height and predict, the chance of predicting a man is 18%. By guessing its just 50%.

## 3.2 Terminology and Notation

We will be mixing the terminology of statistics and computer science. For example, we will sometimes call $Y$ and $X$ the outcome/predictors, sometimes observed/covariates, and even input/output.

We will denote the predictors with $X$ and the outcomes with $Y$ (quantitative) and $G$ (qualitative). Notice $G$ are not numbers, so we can add and multiply them.

Note in the example in the first Section we used $Y$ instead of $G$.

Height and weight are *quantitative measurements*. These are sometimes called continuous measurements.

Gender is a *qualitative measurement*. They are also called categorical or discrete. This is a particularly simple example because there are only two values. With two values we sometimes call it *binary*. We will use $\mathcal{G}$ to denote the set of possible values. For gender it would be $\mathcal{G} = \{\text{Male}, \text{Female}\}$. A special case of qualitative variables are *ordered qualitative* where the one can impose an order. With men/women this can't be done, but with, say, $\mathcal{G} = \{$low, medium, high$\}$ it can.

For both types of variable it makes sense to use the inputs to predict the output. Statisticians call the prediction task *regression* when the outcome is quantitative and *classification* when we predict qualitative outcomes. We will see that these have a lot in a common and that they can be viewed as a task of function approximation (as with the scatter plots).

Notice that inputs also vary in measurement type.

### 3.2.1 Technical Notation

We will follow the notation of the Hastie, Tibshirani, and Friedman book. Observed values will be denoted in lower case. So $x_i$ means the $i$th observation of the random variable $X$. Matrices are represented with bold face upper case. For example $\mathbf{X}$ will represent all observed predictors. $N$ will usually be the number of observations, or length of $Y$. $i$ will be used to denote which observation and $j$ to denote which covariate or predictor. Vectors will not be bold, for example $x_i$ may mean all predictors for subject $i$, unless it is the vector of a particular predictor for all subjects $\mathbf{x}_j$. All vectors are assumed to be column vectors, so the $i-th$ row of $\mathbf{X}$ will be $x_i'$, i.e. the transpose of $x_i$.

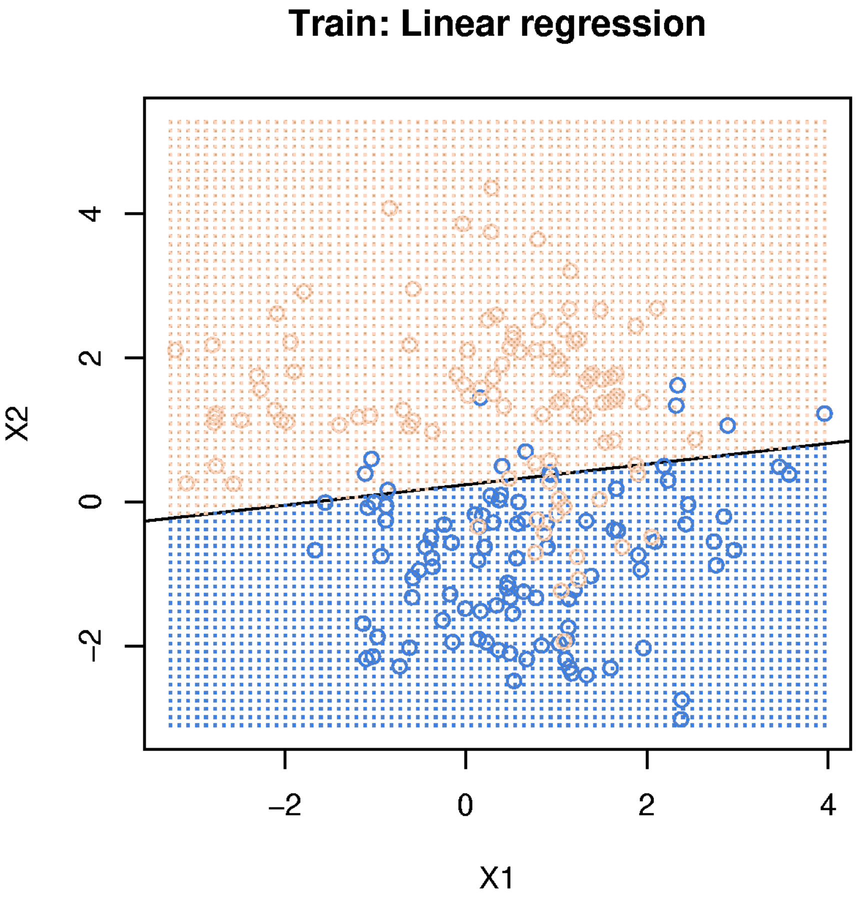
## 3.3 Two covariate example

We turn our attention to a problem where the observations $\mathbf{G}$ and $\mathbf{X}$ where generated from a simulation. There are two covariates so $\mathbf{X}$ has two columns. $N = 200$. Figure 3.3 shows the data:

We will use the hat to denote predictions or estimates. For example our prediction of $Y$ will be denotes with $\hat{Y}$.

The outcome here is binary $\mathcal{G}$ =(Orange, Blue). We will code this with $Y$ =

Figure 3.4: Two predictor example data.



$(0, 1)$.

We will compare two simple, but commonly used methods for predicting $G$ from $X$.

### 3.3.1 Linear Regression

Probably the most used method in statistics. In this case, we predict the output $Y$ via the model

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2$$

However, we do not know what $\beta_0$, $\beta_1$ and $\beta_2$ are.

We use the training data, shown in Figure 3.3 to *estimate* them. We can also say we train the model on the data to get numeric coefficients. We will use the hat to denote the estimates: $\hat{\beta}_1$ and $\hat{\beta}_2$.

We will start using $\boldsymbol{\beta}$ to denote the vector $(\beta_0, \beta_1, \beta_2)'$. A statistician would call these the parameters of the model.

The most common way to estimates $\beta$s is by least squares. In this case we choose the $\boldsymbol{\beta}$ that minimizes

$$\text{RSS}(\boldsymbol{\beta}) = \sum_{i=1}^{N} \{y_i - (\beta_0 + \beta_1 X_1 + \beta_2 X_2)\}^2$$

If you know linear algebra and calculus you can show that $\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$.

Notice we can predict $Y$ for any $X$:

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2$$

To predict the class we could simply say: $\hat{G} =$orange if $\hat{Y} < 0.5$ and blue otherwise. Figure 3.3.1 shows the prediction graphically.

Not bad. We get a prediction error of 14%. Much better than guessing.

Notice that all prediction from linear regression will results in a division of the plane such as that shown in Figure 3.3.1. We call the line that divides the space into orange calls and blue calls the *decision boundary*. The data seems to suggest we could do better by considering more flexible models.

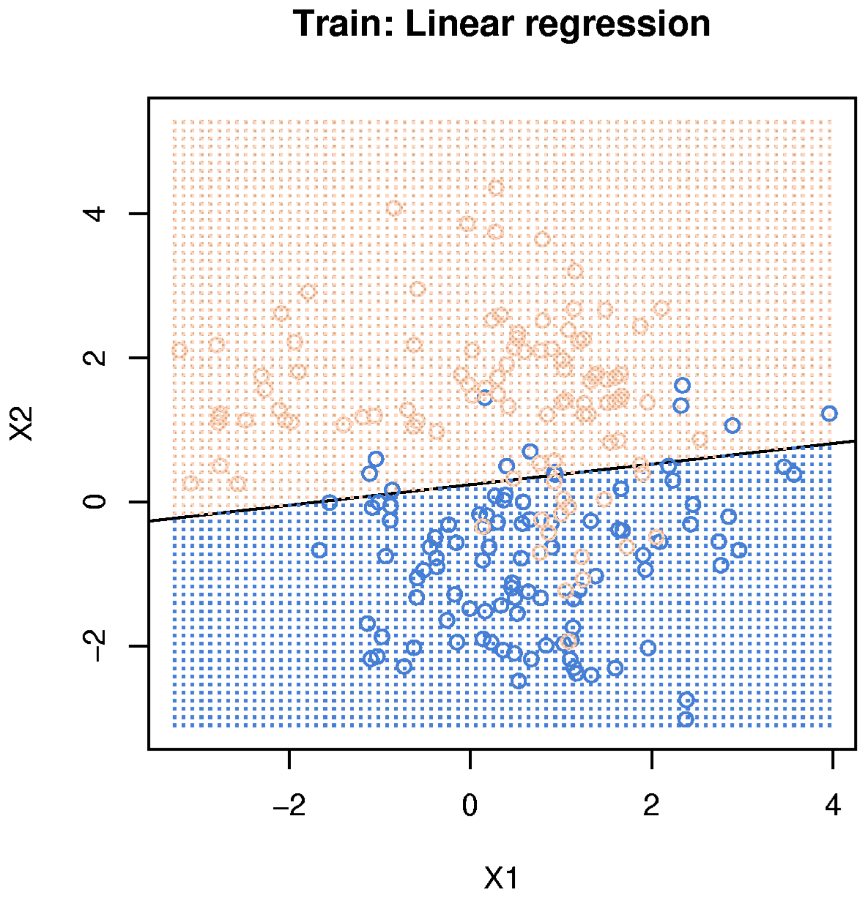
Note: If we don't like the fact that $\hat{Y}$ can be larger than 1 and smaller than 0 we could have used logistic regression. However, for prediction purposes the answer would have been very similar.

### 3.3.2 Nearest Neighbor

Nearest neighbor methods use the points closest in predictor space to $x$ to obtain an estimate of $Y$. For the K Nearest Neighbor method (KNN) we define

$$\hat{Y} = \frac{1}{k} \sum_{x_k \in N_k(x)} y_i.$$

Figure 3.5: Prediction from linear regression.



Here $N_k(x)$ contains the $k$-nearest points to $x$. As before, we can predict using the following *rule*: if $\hat{Y} < 0.5$ we say $G$ is orange and blue otherwise.

Notice, as for regression, we can predict $Y$ for any $X$.

In Figure 3.3.2 we see the results of KNN using the 15 nearest neighbors. The decision boundary seems to adapt better to data and we are able to predict with an error rate of 12%.

We do a bit better with KNN than with linear regression. However, here we have to be careful about *over training*

Roughly speaking, over training is when you mold an algorithm to work very well (sometimes perfect) on a particular data set forgetting that it is the outcome of a random process and our trained algorithm may not do as well in other instance.

Figure 3.3.2 demonstrates what happens when we use KNN with $k = 1$. Our prediction error is 0. Do we really believe we can always do this well with this decision boundary?

It turns out we have been hiding a *test* data set. Now we can see which of these trained algorithms performs best on an independent data set generated by the same stochastic process. Figure 3.3.2 shows the test data.

Notice that the prediction error are worst now. Especially for the KNN using $k = 1$. The small little regions we carved out to predict lonely blue and orange points no longer predict correctly.

Figure 3.6: Prediction from nearest neighbor.

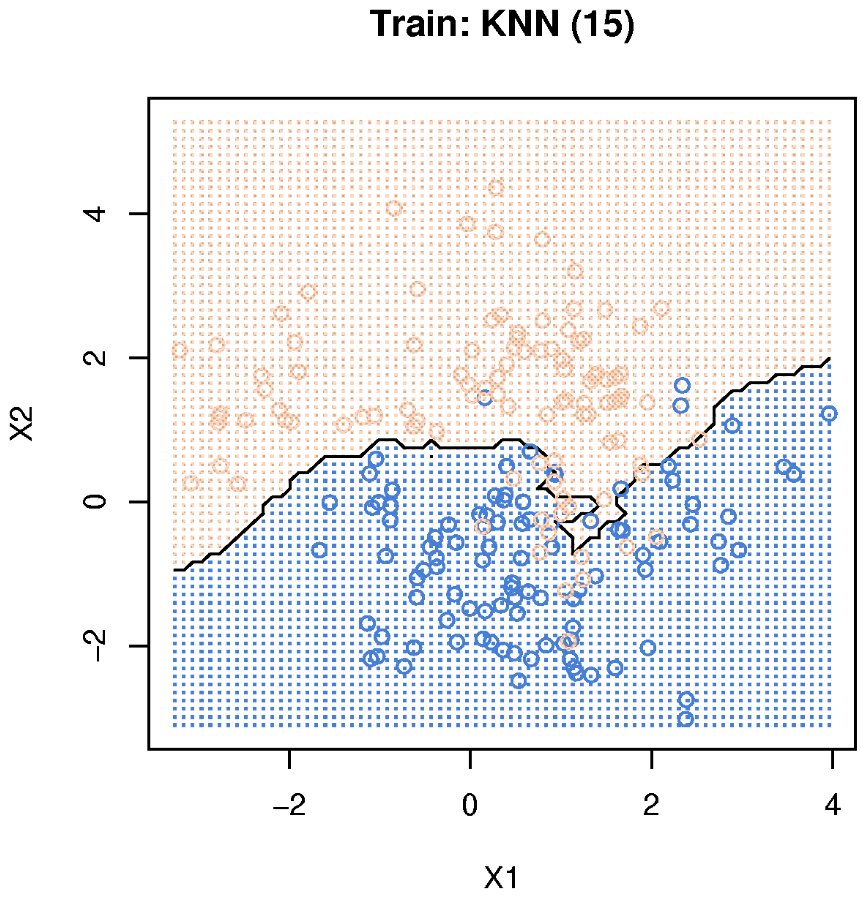


Figure 3.7: Prediction from nearest neighbor.

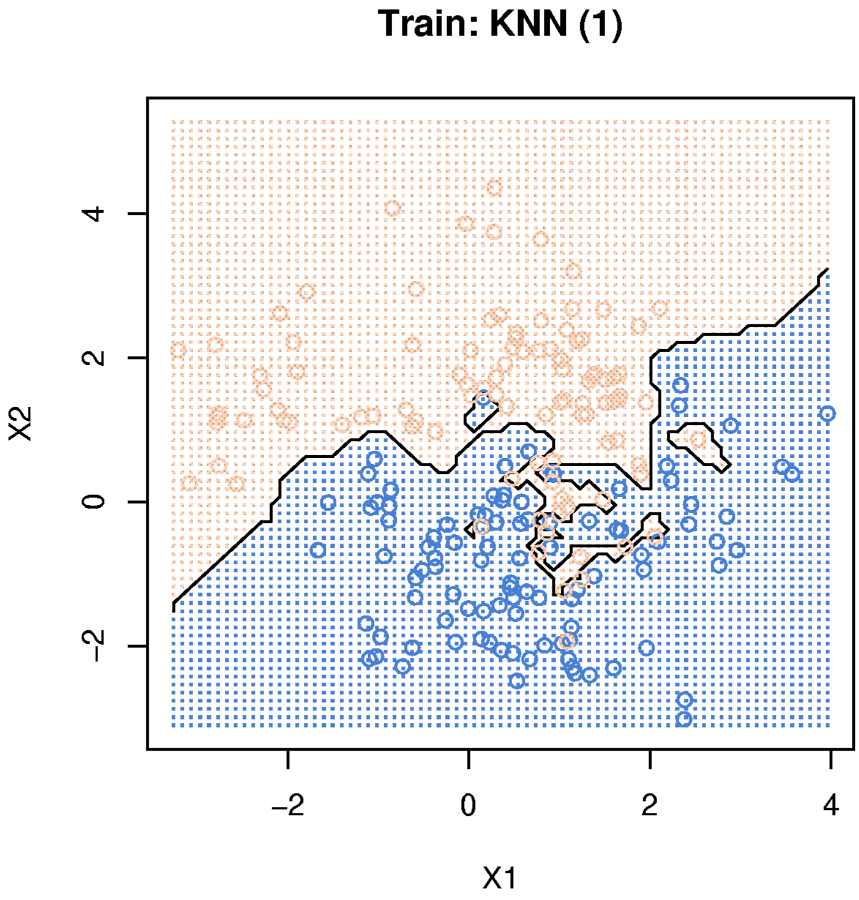
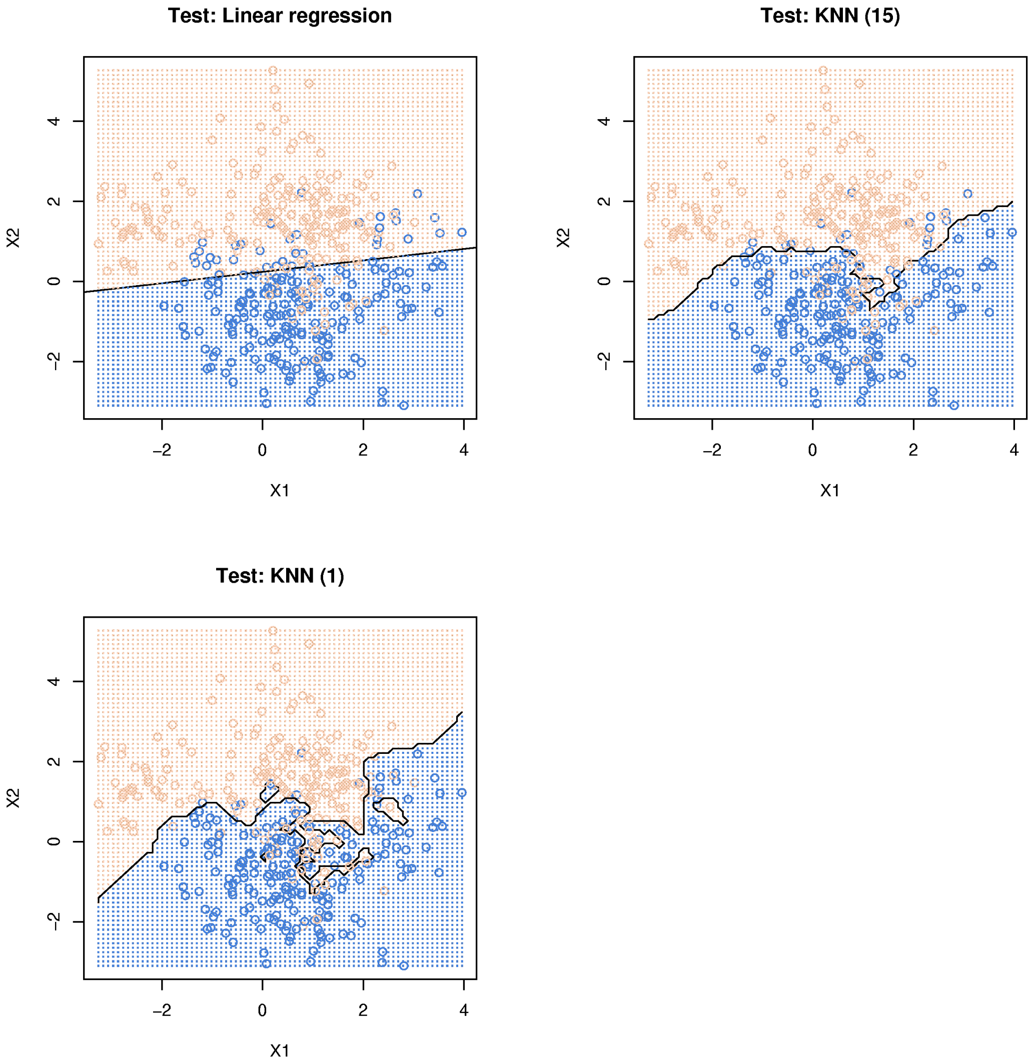


Figure 3.8: Prediction from linear regression and nearest neighbor on test set.



Test: Linear regression

Test: KNN (15)

Test: KNN (1)

| | Prediction error | |
|---|---|---|
| Method | Train | Test |
| Linear Regression | 0.140 | 0.185 |
| KNN(1) | 0.00 | 0.185 |
| KNN(15) | 0.120 | 0.175 |

The following table summarizes the results:

So how do we choose $k$? We will study various ways. First lets talk about the bias/variance or accuracy/precision trade-off.

Smaller $k$ give more flexible estimates, but too much flexibility can results in over-fitting and thus estimates with more variance. Larger $k$ will give more stable estimates but may not be flexible enough. Not being flexible is related to being biased or inaccurate.

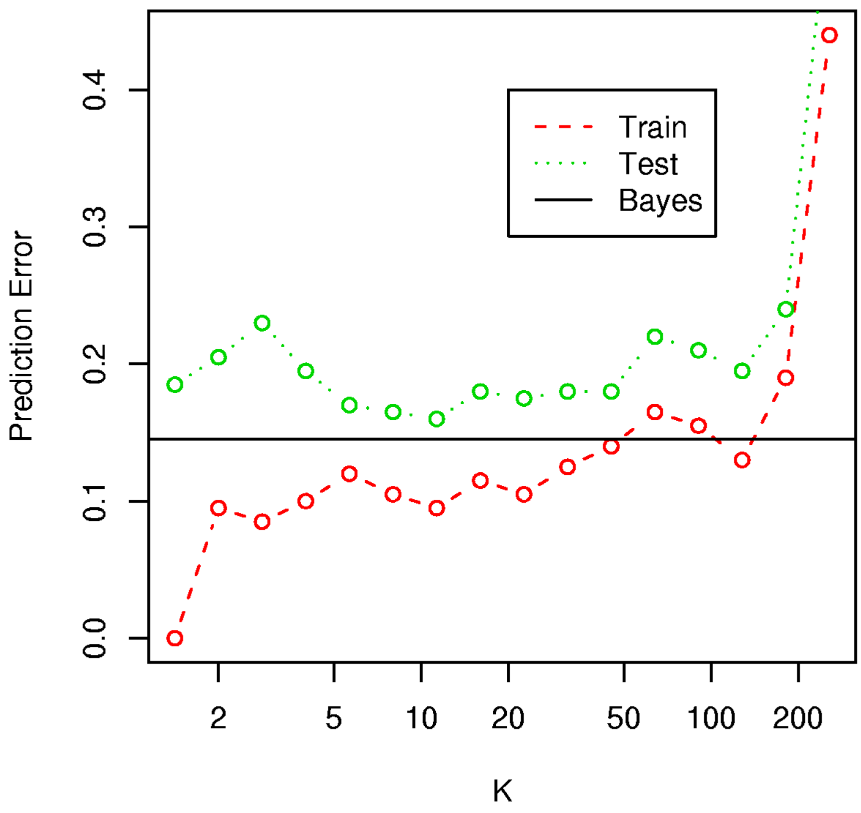
Figure 3.3.2 shows the error rates in the test and training sets for KNN with varying $k$.

Notice that for small $k$ we are clearly over-training.

## 3.4 Bayes Classifier

Can the prediction rule ever be perfect? When their is an underlying stochastic model, understanding Bayes theorems not only tells us that perfection is impossi-

Figure 3.9: Prediction from nearest neighbor.



ble but it also lets us calculate the best we can do.

Recall the expected prediction error (EPE). What do we do for qualitative data? Because the elements of $\mathcal{G}$ are not numbers taking expectation does not mean anything. However, we can define a loss function. Say we have three elements $\mathcal{G} = \{A, B, C\}$ we can define the loss function like this:

$$L(\hat{G}, G) = \begin{cases} 0 \text{ if } \hat{G} = G \\ 1 \text{ if } \hat{G} \neq G \end{cases}$$

This can be thought of a distance matrix with 0s in the diagonals and 1s everywhere else.

Notice in some fields this loss function may not be appropriate. For example saying someone has cancer when they do not may not be as bad as saying they do not have cancer when they do. For now, lets stick to this 0 and 1 case.

We can now write:

$$\text{EPE} = \mathrm{E}_X \sum_{k=1}^{K} L[\mathcal{G}_k, \hat{G}(X)] \, \mathrm{Pr}(\mathcal{G}_k | X)$$

The solution to this is what is know as *Bayes classifier*.

$$\hat{G}(X) = \max_{g \in \mathcal{G}} \mathrm{Pr}(g | X = x)$$

So why don’t we use it?

Typically we do not know $\Pr(g|X = x)$. Just like we didn't know $f(x) = E[Y|X = x]$.

However, in the examples shown above I now the random mechanism that created the data. I wrote the code! But, remember that in practice we rarely know this! However, in my illustrative examples I I can calculate $\Pr(g|X = x)$ and can create the Bayes classifier, seen in Figure 3.4 shows.

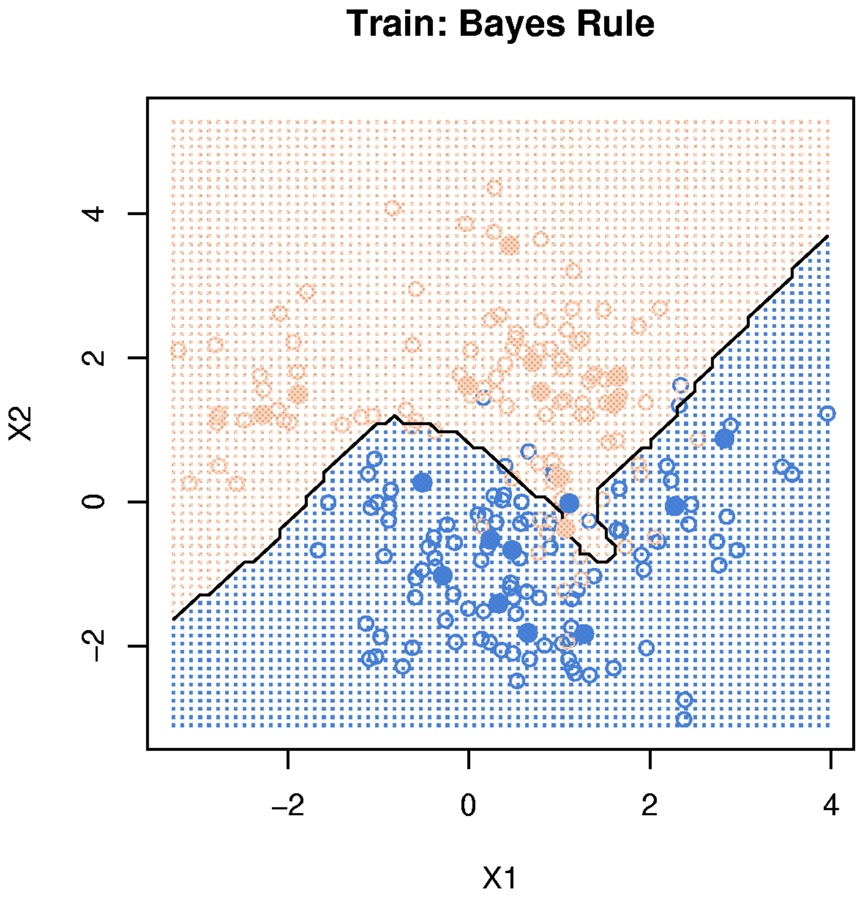
Figure 3.4 shows the clusters of subpopulations used to create the data. Basically we took bivariate normal random variable with the means shown as large dots.

Notice that KNN is an intuitive estimator of the Bayes classifier. We do not know what the function $\Pr(G = g|X = x)$ looks like so we estimate it with the number of $g$s in a small area around $x$. The more neighbors I consider the larger the area.

Technical note: If we code $Y = 1$ if $G = g$ and $Y = 0$ otherwise then we can see the relationship between Bayes classifier and the regression function because: $\Pr(G = g|X = x) = E(Y|X = x)$

Instead of KNN we can use a *kernel smoother* to estimate $Pr(g|X = x)$. We will talk about these later. A very famous classifier (support vector machine) has lots in common with a kernel smoother.

Figure 3.10: Bayes classifier



## 3.5 Curse of dimensionality

So why would we ever consider a method such as linear regression? Is a KNN and kernel smoother 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 multidimensional "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 $l \times l \times \ldots \times l = l^p = .10$. 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. We will talk about CART later in the term.

## 3.6 Function Approximation

Let's move our attention to continuous data.

As mentioned previously the best predictor for Y given X is is the regression function

$$f(x) = \mathrm{E}(Y|X = x)$$

To be able to predict we need to know $f$. In most situations we do not know $f$.

In some cases *science* tells us what $f$ should be. For example, Hook's law tells us that if we are measuring the amount a spring stretches $Y$ when holding a weight $X$ then

$$f(x) = \beta_0 + \beta_1 x$$

$\beta_0$ is the original length of the spring and $\beta_1$ is known as the *spring constant*. For any given spring we don't know the constant but by measuring various weights we can figure it out.

Measurement error will make it a bit tricky. A stochastic model is

$$Y = f(X) + \epsilon$$

where $\epsilon$ is an unbiased measurement error that does not depend on $X$.

With enough data least squares will give us very precise estimates and we will be able to predict the stretch accurately and precisely.

Note: There are also examples where science suggest non-linear models. For example, some physical chemistry results suggest one use sigmoidal models. We can use least squares here two although computationally its harder.

In the case of Weight and Height is there science telling us the relationship is linear? Not really. However, much empirical evidence shows they are bivariate normal with correlation of about $0.5$ which implies the relationship is linear as well.

There are many situations where neither science nor empirical evidence suggest a linear model. Despite this, the norm in many fields seems to be to use linear regression first and *not* ask questions later.

Homework: Take the best journal in your field. Find a paper that uses linear regression. Do the authors present evidence, based on deductive reasoning, supporting a linear relationship? Do they present empirical evidence?

A common approach is to define a family of functions to choose from. Then using, for example, least squares, pick the function in this family that best fits the data. We then have 2 ways we can screw up: 1) the true $f$ may not be well represented by any of the functions in the family we choose and 2) our estimate of the best functions in the family is not very good.

The discrete data examples in the previous sections show this.

An example of 1) is the use of linear models to classify the orange and blue in the sections above. The best line is not good enough. We refer to this as bias.

An example of 2) is the use of KNN with $k = 1$. In this case the family of possible functions is includes $f(x)$ but our estimate is not very good because its based on few data points (just 1 per location!). We call this variance.

## 3.7 Introduction to the Bias-Variance Trade-off

Note: This sections is very mathematical.

In general, smoothing problems present a fundamental trade-off between bias and variance of the estimate, and this trade-off is governed by some tuning parameter, e.g. k in KNN. In this Section we introduce the general problem of function approximation, with continuous outcomes, and demonstrate a way to quantify the bias/variance trade-off.

Through out this section we will be using an artificial example defined by the following simulation

$$y_i = 5\sin(1/x) + \epsilon_i, i = 1, \dots, n \tag{3.1}$$

with the $\epsilon_i$ IID $N(0,1)$ or $t_3$.

The trade-off is most easily seen in the case of the running mean smoother (just like KNN). The fitted running-mean smooth can be written as

$$\hat{f}_k(x_0) = \frac{1}{2k+1} \sum_{i \in N_k^S(x_0)} y_i$$

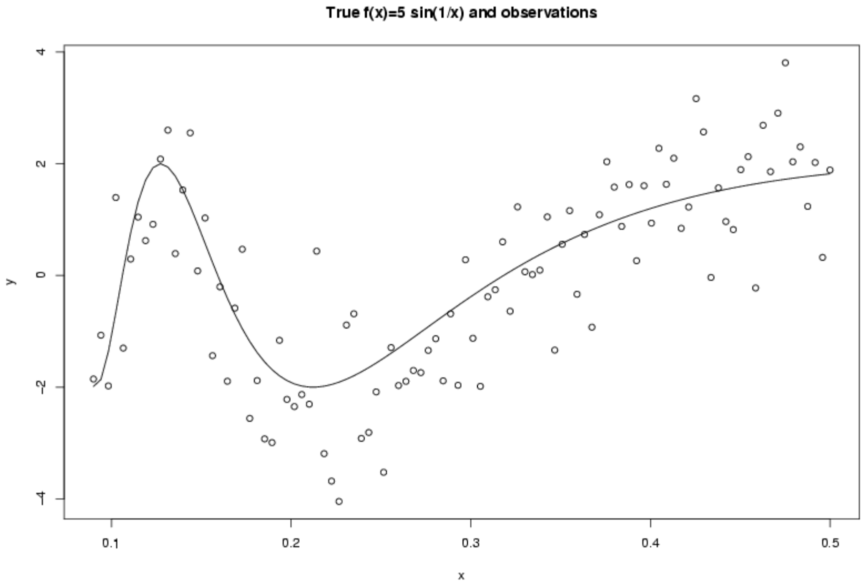


Figure 3.11: Outcomes of model with $f(x) = 5\sin(1/x)$ and IID normal errors with $\sigma^2 = 1$

The variance is easy to compute. What is it?

The bias is

$$\mathrm{E}[\hat{f}_k(x_0)] - f(x_0) = \frac{1}{2k+1} \sum_{i \in N_k^S(x_0)} \{f(x_i) - f(x_0)\}$$

Notice that as $k$, in this case the smoothing parameter, grows the variances decreases. However, the bigger the $k$ the more $f(x_i)$'s get into the bias.

We have no idea of what $\sum_{i \in N_k^S(x_0)} f(x_i)$ is because we don't know $f$! Let's see this in a more precise (not much more) way.

Here we know $f$ but in general we do not. Say we think that $f$ is smooth enough for us to assume that its second derivative $f''(x_0)$ is bounded. Taylor's theorem says we can write

$$f(x_i) = f(x_0) + f'(x_0)(x_i - x_0) + \frac{1}{2} f''(x_0)(x_i - x_0)^2 + o(|x_i - x_0|^2).$$

Because $\frac{1}{2} f''(x_0)(x_i - x_0)^2$ is $O(|x_i - x_0|^2)$ we stop being precise and write

$$f(x_i) \approx f(x_0) + f'(x_0)(x_i - x_0) + \frac{1}{2} f''(x_0)(x_i - x_0)^2.$$

Implicit here is the assumption that $|x_i - x_0|$ is small. This is the way these asymptotics work. We assume that the kernel size goes to 0 as $n$ gets big.

Why did we only go up to the second derivative?

To makes things simple, let's assume that the covariates $x$ are *equally spaced* and let $\Delta = x_{j+1} - x_j$ we can write

$$(2k+1)^{-1} \sum_{i \in N_k^S(x_0)} f(x_i) \approx f(x_0) + (2k+1)^{-1} \frac{k(k+1)}{6} f''(x_0)\Delta^2$$

So now we see that the bias increases with $k^2$ and the second derivative of the "true" function $f$. This agrees with our intuition.

Now that we have

$$\mathrm{E}\{\hat{f}_k(x_0) - f(x_0)\}^2 \approx \frac{\sigma^2}{2k+1} + \frac{k(k+1)}{6} f''(x_0)\Delta^2$$

we can actually find an optimal $k$

$$k_{opt} = \left\{ \frac{9\sigma^2}{2\Delta^4\{f''(x_i)\}^2} \right\}$$

Usually this is not useful in practice because we have no idea of what $f''(x)$ is like. So how do we chose smoothing parameters?

In Figure 3.12 we show the smooths obtained with a running mean smoother with bandwidths of 0.01 and 0.1 on 25 replicates defined by (3.1). The bias-variance trade-off can be clearly seen.

Through-out the class we will be discussing performance in the context of this trade-off.

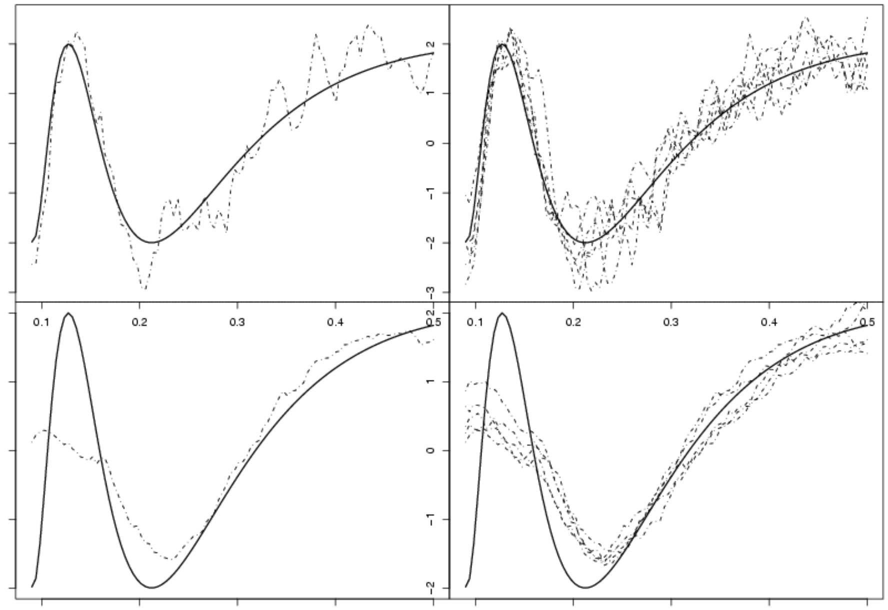


Figure 3.12: Smooths using running-mean smoother with bandwidths of .01 and 0.1. To the right are the smooths 25 replicates