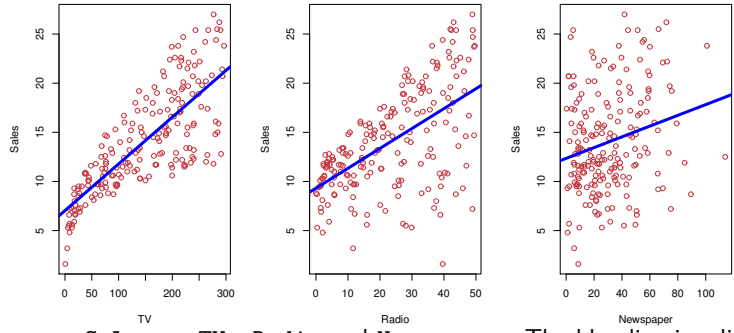
STAT318/462 — Data Mining

Dr Gábor Erdélyi

University of Canterbury, Christchurch, New Zealand

Course developed by Dr B. Robertson. Some of the figures in this presentation are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.

What is Statistical Learning?



- Shown are `Sales` vs. `TV`, `Radio` and `Newspaper`. The blue line is a linear regression fit.
- **Question:** Can we predict `Sales` using these three predictors?

There is a positive relationship between each predictor and sales - an increase in X tends to correspond to an increase in Y. There is greater variability in sales for the predictor Newspaper. Heteroscedasticity (non-constant variability in Sales) is visible for the predictors TV and Radio (the variability in Y increases as X increases).

Revision material (for next slide): A random variable X is a mapping from the sample space to the set of real numbers, where each distinct outcome is assigned a unique value. A coin toss, for example, has two outcomes (Head,Tail) and a (**discrete**) random variable could be $X = 1$ if Head and $X = -1$ if Tail. A probability density function (PDF), $g(x)$, is a function that describes the probability of a **continuous** random variable X falling within a particular range:

$$\begin{aligned} g(x) &\geq 0 \text{ for all } x \\ \Pr(a \leq X \leq b) &= \int_a^b g(x)dx \\ \int_{-\infty}^{\infty} g(x)dx &= 1 \quad \text{(area under the curve is one)} \end{aligned}$$

A well-known PDF is the Normal distribution. We will not be evaluating integrals in this course, but you should be familiar with the concept of a PDF.

- Suppose we have a **quantitative** response Y (regression problem) and p different predictors,
$$X = (X_1, X_2, \ldots, X_p).$$
- We assume there is a relationship between Y and X of the form
$$Y = f(X) + \epsilon,$$
where ϵ is a random error term which is independent of X and has mean $E(\epsilon) = 0$.
- Essentially, we are interested in approaches for estimating f (statistical learning).

The function $f(X)$ will never model Y perfectly. The error ϵ captures measurement errors and all other discrepancies (for example, unmeasured variables that also contribute to Y).

Revision Material: Let X be a random variable. The expected value of X (*the long run average value of repeated draws of* X) is

$$E(X) = \int_{-\infty}^{\infty} x g(x) dx, \text{ where } g(x) \text{ is the PDF for } X \text{ (**continuous** } X)$$

$$E(X) = \sum_{x} x \Pr(X = x), \text{ where } \Pr(X = x) \text{ is the probability that } X = x \text{ (**discrete** } X)$$

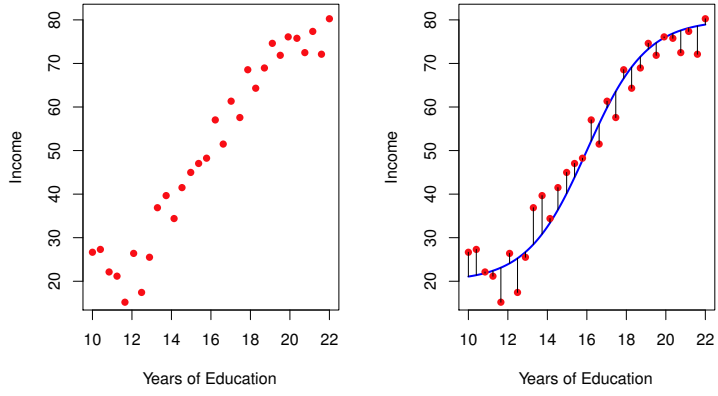
Useful properties for this course:

$$\begin{aligned} E(a) &= a \quad (\text{where } a \text{ is a number}) \\ E(aX) &= aE(X) \\ E(X_1 + X_2) &= E(X_1) + E(X_2) \end{aligned}$$

The variance of X (*how far a set of X's are spread out from their mean*) is

$$\begin{aligned} V(X) &= E[(X - E(X))^2] = E(X^2) - (E(X))^2 \\ V(aX) &= a^2 V(X) \end{aligned}$$

Simulated Example



- Shown are `Income` vs. `Years of Education`. The blue line is the true underlying relationship, $f(X)$.

The true underlying relationship, $f(X)$, is a non-linear function. The red dots are observed (measured) values of

$$(x, y) = (\text{Years of Education, Income}),$$

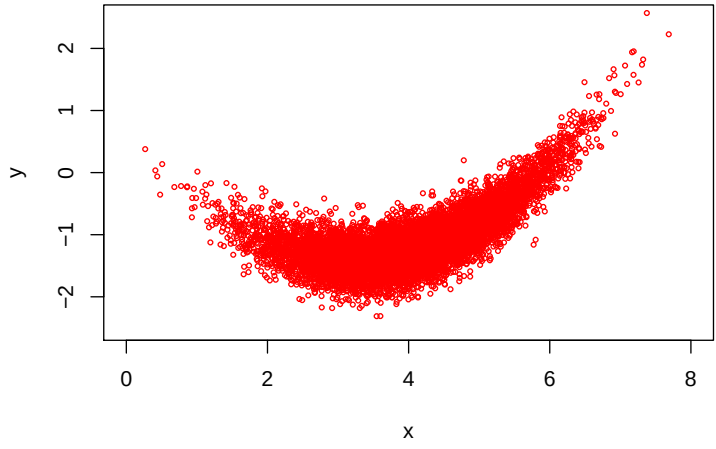
drawn from $Y = f(X) + \epsilon$. The black lines denote the errors (the difference between an observed value of y at $X = x$ and $f(x)$) which are positive (y is above $f(X)$) and negative (y is below $f(X)$) with a mean of approximately zero.

What is $f(X)$ good for?

- We can use a good f to make predictions of Y at unseen test cases $X = \mathbf{x}$.
- We can determine which predictors are important in explaining Y, and those that are not.
- Potentially understand how each predictor affects Y.

Remember that it's usually relatively easy to measure the predictors and difficult or impossible to measure the response directly. Hence, making predictions at unseen test cases is very useful. Making inferences about the predictors is also important. For example, we would expect *Seniority* and *Years of Education* to have a positive affect on *Income*, but another variable like *Marital Status* may have no affect. We want to learn which predictors are important for predicting the response from our observed data, particularly if we have many predictor variables (sometimes called wide data).

Is there an ideal $f(X)$?



- There can be many Y values at $X = \mathbf{x}$.

In this example, I simulated 50,000 data pairs (x, y) from a specified distribution.

- Based on these data, is there a good value for $f(X)$ at $X = x$?
- There are many values of Y for a given x, but a function can only give one value. Is there a *best* value?

Is there an ideal $f(X)$?

- The ideal $f(X)$ at $X = \mathbf{x}$ is

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

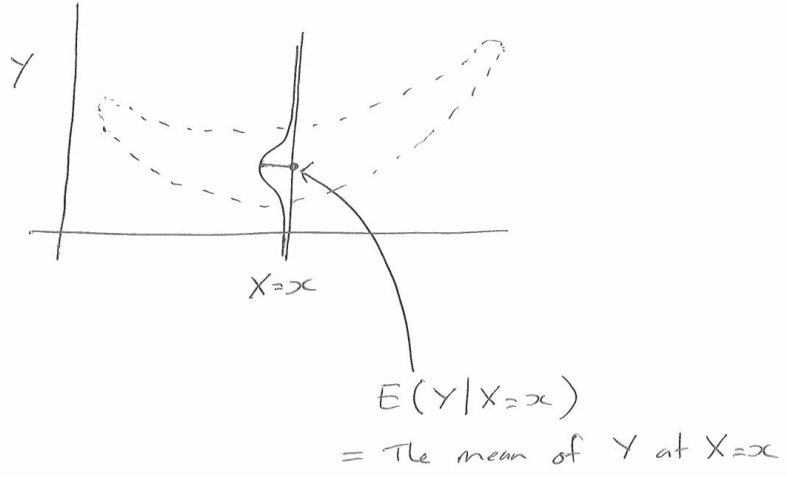
and is called the **regression function**.

- This function is ideal with respect to mean-squared prediction error ($E(Y - f(X))^2$). To be specific, it minimizes

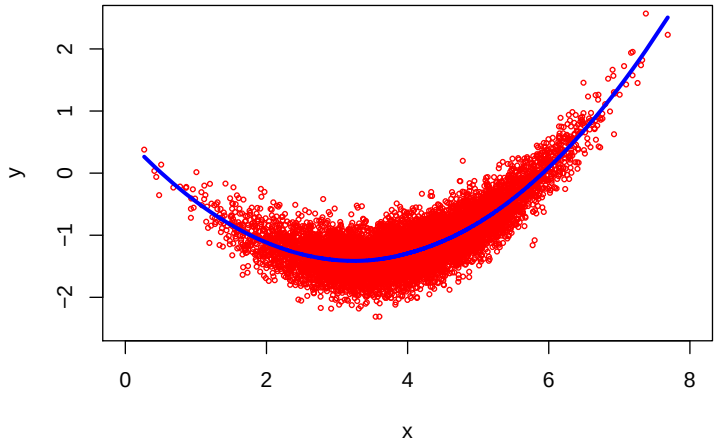
$$E\left[(Y - g(X))^2|X = \mathbf{x}\right]$$

over all functions g at all points $X = \mathbf{x}$.

For convenience, let's assume that Y is normally distributed at $X = x$ (it's easier to draw). The conditional expectation of Y at a particular x is sketched below.



The ideal $f(X)$



- The regression function, $f(\mathbf{x}) = E(Y|X = \mathbf{x})$.

In this example we can plot the regression function because we know the population's distribution. In practice, we do not know enough about the population to compute the regression function. We need to estimate the regression function using observed data and there are many possible ways of doing this.

The ideal $f(X)$

- Even if we knew $f(\mathbf{x})$, we would still make errors in prediction because there is typically a distribution of Y values at each $X = \mathbf{x}$.
- This error is called **irreducible error**, given by

$$\epsilon = Y - f(\mathbf{x}).$$

- Suppose we have an estimate $\hat{f}(\mathbf{x})$ for $f(\mathbf{x})$, then we can decompose the average mean-squared error as follows:

$$E\left[(Y - \hat{f}(X))^2 | X = \mathbf{x}\right] = \underbrace{[f(\mathbf{x}) - \hat{f}(\mathbf{x})]^2}_{\text{(reducible)}} + \underbrace{V(\epsilon)}_{\text{(irreducible)}}$$

On average $f(X)$ does well, but we will always make errors.

Assume, for the moment, that $\hat{f}$ is fixed and $X = x$, then

$$\begin{aligned} E((Y - \hat{f}(x))^2) &= E((f(x) + \epsilon - \hat{f}(x))^2) \\ &= E([f(x) - \hat{f}(x)]^2 + 2\epsilon(f(x) - \hat{f}(x)) + \epsilon^2) \\ &= [f(x) - \hat{f}(x)]^2 + 2E(\epsilon)(f(x) - \hat{f}(x)) + E(\epsilon^2) \\ &= [f(x) - \hat{f}(x)]^2 + V(\epsilon) \end{aligned}$$

(Note: $E(\epsilon^2) = E((\epsilon - E(\epsilon))^2) = V(\epsilon)$ because $E(\epsilon) = 0$)

The irreducible error ($V(\epsilon)$) is fixed and we are stuck with it (regardless of how fancy our model is!). We have some control over the reducible error ($[f(x) - \hat{f}(x)]^2$). We will consider different ways of estimating f with the aim of minimizing the reducible error.

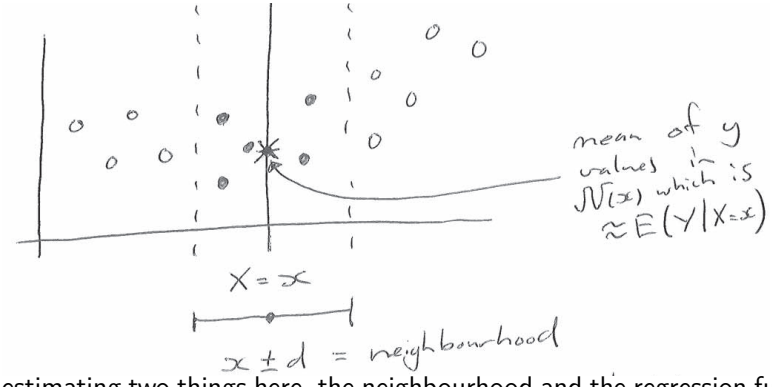
How do we estimate $f(\mathbf{x})$?

- We cannot compute $E(Y|X = \mathbf{x})$ directly because we don't know the conditional distribution of Y given X.
- At best, we have a few observations of Y near $X = \mathbf{x}$.
- One approach is to relax the definition and let

$$\hat{f}(\mathbf{x}) = \text{Ave}(Y|X \in \mathcal{N}(\mathbf{x})),$$

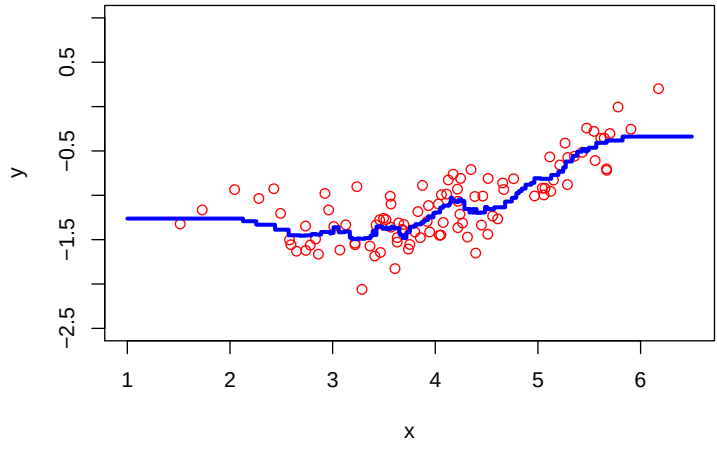
where $\mathcal{N}(\mathbf{x})$ is a neighbourhood of $\mathbf{x}$.

We usually don't have many (or any) Y values at $X = x$, but we do have some values in a neighbourhood of x. If the neighbourhood is relatively small and it contains many observations, the neighbourhood's mean will be a good approximation to $f(x) = E(Y|X = x)$. A simple example is illustrated below (shaded points are in $\mathcal{N}(\mathbf{x})$).



We are estimating two things here, the neighbourhood and the regression function.

Nearest neighbour averaging



- A sample of $n = 100$ (averaging using 10 neighbours).

In class *R* demonstration.

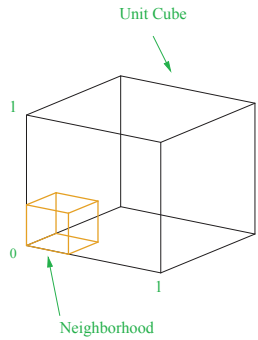
This method is called *k*-nearest neighbour regression, where the neighbourhood is defined as the k nearest neighbours (in terms of Euclidean distance here). For larger values of k, $\hat{f}(x)$ is *smoother* (less jumpy) than it is for smaller values of k. The method is useful for interpolation (making predictions within the range of the data), but not for extrapolation (outside the range of the data). The performance of this method depends heavily on k. If k is too large, you will fail to capture the underlying structure in the data (called under-fitting). If k is too small, you will start following the errors (the ϵ term in $Y = f(X) + \epsilon$) too closely (called over-fitting) and the predictions will not be stable — a small change in x can correspond to a large change in y. You will learn how to choose the *best* k value later in the course using the method of cross-validation.

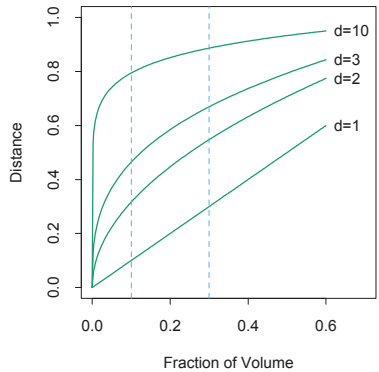
Nearest neighbour averaging

- Problem solved, we have a good estimate for the regression function! **Or do we?**
  1. If p (dimension) is small and n (sample size) is large, nearest neighbour averaging works reasonably well!
  2. However, smoother versions exist in this case (e.g. splines).
  3. If p is large, nearest neighbour averaging tends to perform poorly because 'nearest neighbours' tend to be far away. This problem is called the **curse of dimensionality**.

We need to have local information to obtain a good approximation to $E(Y|X = x)$ using nearest neighbour averaging. If too few points are close to x, we tend to get poor local estimates with large variance.

Curse of dimensionality





- Sub-cube neighbourhood in the unit cube.

To illustrate the curse of dimensionality, let's define each neighbourhood as 10% of the volume of the unit cube.

- If $d = 1$, the interval $[0, 0.1]$ is 10% of the volume.
- If $d = 2$, the box $[0, 0.32] \times [0, 0.32]$ is 10% of the volume.
- If $d = 10$, the box $[0, 0.79]^{10}$ is 10% of the volume ($\approx 80\%$ of the range of each dimension is included in this box).

Hence, these 10% neighbourhoods may no longer be local in high dimensions. This loses the spirit of estimating $E(Y|X = x)$ using local averaging. The take home message here is that it is hard to find near neighbours and stay local in high dimensions. If this problem didn't exist, we would always use local averaging methods.

Another way to illustrate the curse of dimensionality uses sampling density. If $N_1 = 10$ points represents a dense sample with one predictor, we would require $N_2 = 100$ points for two predictors and $N_{10} = 10^{10}$ points for ten predictors to maintain the same point density. A direct consequence of this is that all feasible training samples sparsely populate the predictor space.

Parametric and Structured Models

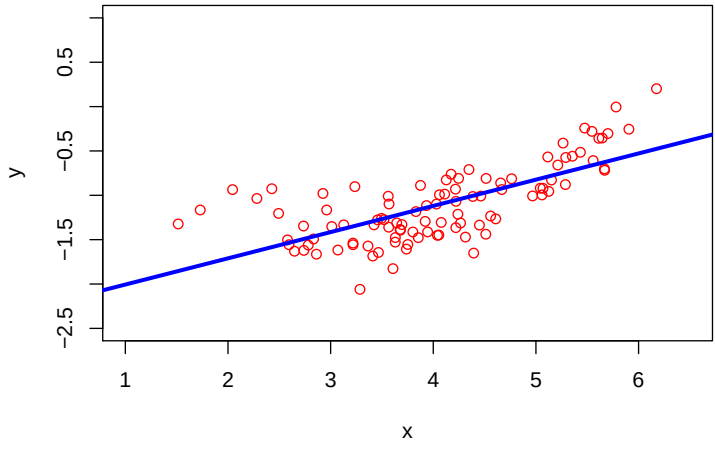
- Make an assumption about the functional form (shape) of f.
- A simple and extremely useful parametric model is the linear model:

$$f(X) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p.$$

1. $p + 1$ parameters: $\beta_0, \beta_1, \ldots, \beta_p$.
2. Estimate the parameters by fitting the model to training data.
3. *"all models are wrong, but some are useful" (George Box).*

One way to move on from local averaging methods is to make assumptions about the functional form of the regression function. Linear regression fits a linear model to the training data and does not use local averaging. The linear model is almost never correct, but it often serves as a good and easy to interpret approximation to $E(Y|X = x)$ (remember that in regression we are always approximating the mean response).

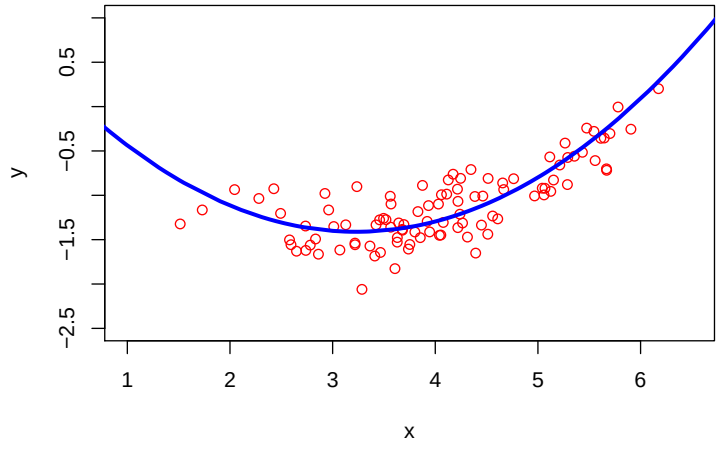
Linear Model



- $\hat{f}(X) = \hat{\beta}_0 + \hat{\beta}_1 X$.

The simple (one predictor) linear model is a reasonable fit here. However, looking at the error terms suggests that the $E(\epsilon) = 0$ assumption is violated here. The hats on the coefficients $\hat{\beta}_0$ and $\hat{\beta}_1$ indicate that they have been estimated from training data.

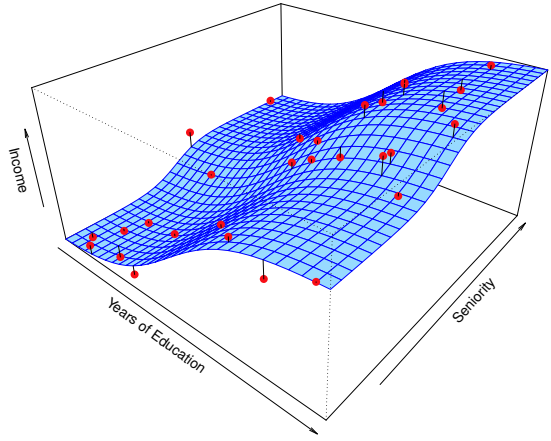
Quadratic Model



- $\hat{f}(X) = \hat{\beta}_0 + \hat{\beta}_1 X + \hat{\beta}_2 X^2$.

Subjectively, the quadratic model fits the training data better than the simple linear model did (and the $E(\epsilon) = 0$ assumption seems reasonable). Although there is a quadratic term (X^2) in the model, it is still a linear model because it is linear in the coefficients (β_0, β_1 and β_2). Geometrically, the model is a plane in the $X \times X^2$ predictor space and it is a quadratic curve in X.

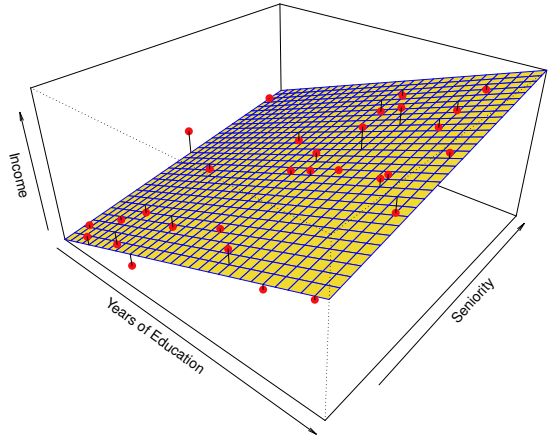
Simulated Model



- $\texttt{Income} = f(\texttt{Education},\texttt{Seniority}) + \epsilon$.

This is simulated data from an artificial population, where the blue surface is the true regression function.

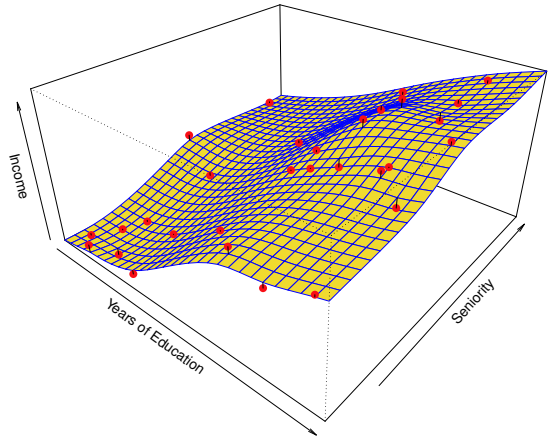
Parametric Model: Linear Model



- $\texttt{Income} \approx \hat{\beta}_0 + \hat{\beta}_1 \texttt{Education} + \hat{\beta}_2 \texttt{Seniority}$.

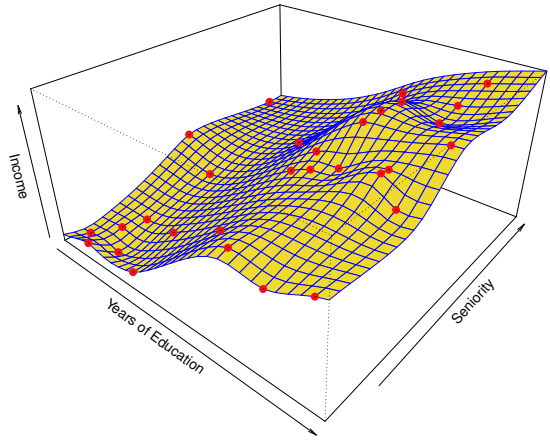
If we assume that the regression function is linear (even though we know from the previous slide that the true regression function is not linear), we can fit a linear model to the training data. The linear model does a reasonable job here and only requires three parameters to be estimated (β_0, β_1 and β_2). The model is simple and very easy to interpret: an increase in years of education tends to increase income etc.

Non-Parametric Model: Thin-Plate Spline



The spline (which is a smooth version of neighbourhood averaging) has done extremely well. This approach does not impose any pre-specified model for $f(X)$. It fits a smooth surface to the training data subject to a 'level of smoothness' parameter (where larger values allow for more wiggly fits). Although splines are considered for illustrative purposes in this section, we will not be studying them in this course (see Chapter 7 of the course text in you're interested). The take home message here is that it is possible to fit a non-linear model to this data and get a better approximation to the regression function.

Non-Parametric Model: Thin-Plate Spline



In this example, the spline's smoothness parameter has been increased (a more wiggly surface) so that the surface passes through all the training data points. That is, the fitted model has **zero training error**:

$$y_i = \hat{f}(x_i) \text{ for all the training pairs } (x_i, y_i).$$

However, we expect the model to have errors because the data was generated with errors:

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

This is called **over-fitting the training data**. Over-fitted models tend to give poor predictions of the response because the model follows the errors too closely.

Some trade-offs

1. Interpretability vs. prediction accuracy.
2. Good-fit vs. over-fit or under-fit.
3. Black-box vs. parsimony.

"Everything should be made as simple as possible, but not simpler." (Einstein)

1. Linear models might not give the best predictions, but they are easy to interpret because the response is simply a linear combination of the predictors. On the other hand, splines are more difficult to interpret because the response is a complicated surface, but they might provide accurate predictions.
2. From slides 18-20, we can make the following hand waiving comments:
   - the linear model slightly under-fits the training data;
   - the first spline fits the training data very well; and
   - the second spline over-fits the training data.
3. Clearly we need to establish some objective ways of choosing the best model for a particular problem. However, generally speaking, we usually would prefer a simple model using few predictors (parsimonious) than a complicated black-box model using many predictors. If a simple model performs as well as a more complicated one, in most cases you should choose the simple model.

Assessing Model Accuracy

- There is no *free lunch* in statistical learning meaning that there is no *best* method for all problems. We need a way of determining how well each method performs.

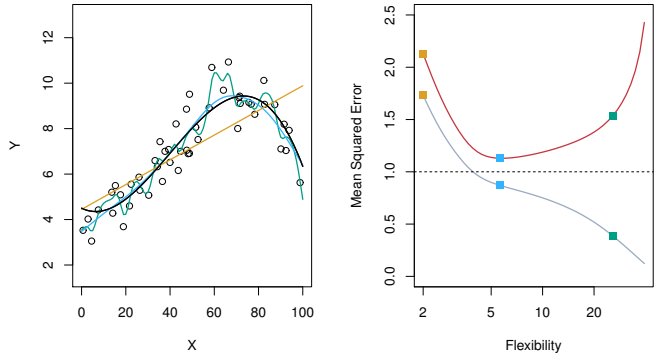
- Suppose we fit a model $\hat{f}(\mathbf{x})$ to some training data $\text{Tr} = \{\mathbf{x}_i, y_i\}_{i=1}^n$. From this set we compute the **training mean-squared error**:

$$\text{MSE}_{\text{Tr}} = \frac{1}{n}\sum_{i=1}^{n}[y_i - \hat{f}(\mathbf{x}_i)]^2.$$

- Suppose we have **unseen** test data $\text{Te} = \{\mathbf{x}_i, y_i\}_{i=1}^m$. From this set we compute the **testing mean-squared error**:

$$\text{MSE}_{\text{Te}} = \frac{1}{m}\sum_{i=1}^{m}[y_i - \hat{f}(\mathbf{x}_i)]^2.$$

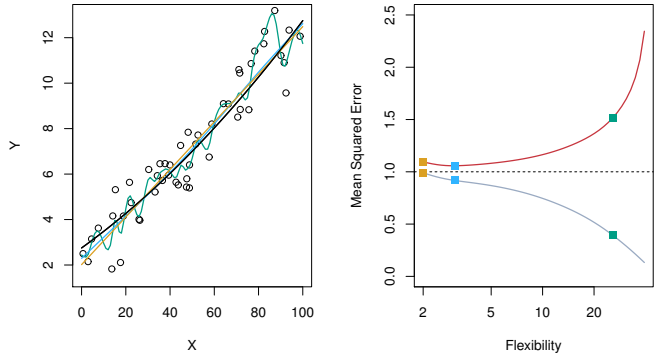
The training MSE is not a good measure of how well a method is doing because it is biased towards models that over-fit the training data. That is, models that over-fit the training data have low training errors. It's easy to create a model with low (or even zero in some cases) training error. The testing MSE gives a much better indication of how well the model is doing because the testing data was not used to build the model. We prefer models with relatively low testing MSEs.



- $f(X)$ is black, MSE_{Tr} is grey, MSE_{Te} is red, and three models are shown in the other colours (yellow is linear, blue and green are splines).

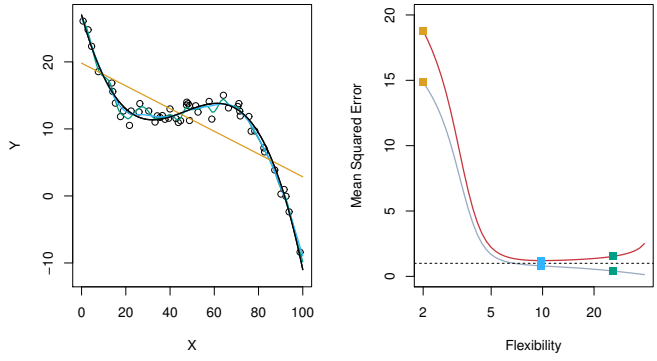
Once again we are using simulated data here, where $Y = f(X) + \epsilon$. Three models have been trained with varying degrees of flexibility (or complexity). The simplest (with two parameters) is a linear model, followed by two splines (with flexibilities 5 and 22). As model flexibility increases, the fits get wigglier and follow the training data more closely. That is, the training error decreases as flexibility increases. However as flexibility increases, the model starts following the errors too closely and over-fitting occurs (the dashed line is the irreducible error, $V(\epsilon)$). We can see that most flexible spine has a training error that is much less than $V(\epsilon)$. Hence, this model is reducing ϵ, which by definition, cannot be predicted by X.

The blue function (the spine with flexibility 5) performed best on this data because its testing MSE was closest to the irreducible error (dashed line).



- $f(X)$ is black, MSE_{Tr} is grey, MSE_{Te} is red, and three models are shown in the other colours (yellow is linear, blue and green are splines).

In this example, the regression function is close to being linear. Hence, the linear model (lowest flexibility) has performed well. In contrast, the green model (the most flexible spine) has over-fitted the training data (the testing MSE is high and the training error is less than the irreducible error).



- $f(X)$ is black, MSE_{Tr} is grey, MSE_{Te} is red, and three models are shown in the other colours (yellow is linear, blue and green are splines).

In this example, the linear model is under-fitting the training data, failing to capture the structure of non-linear regression function (the testing MSE is high and the training MSE is also high). The splines have both performed well in this example, with the blue spline being the best (simpler (less complex) model and less over-fitting).

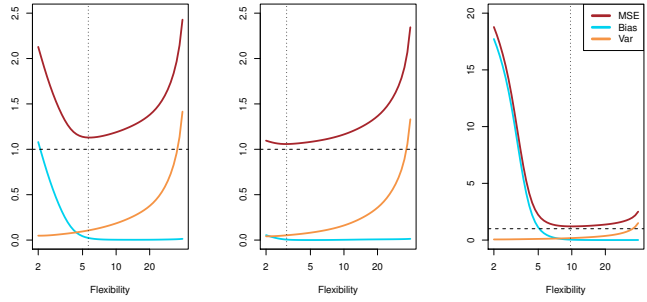
Bias-variance trade-off

- Suppose we fit a model to some training data and let $(\mathbf{x}_0, y_0)$ be a test observation.
- If the true model is
$$Y = f(X) + \epsilon,$$
with regression function $f(\mathbf{x}) = E(Y|X = \mathbf{x})$, then the expected test MSE at $X = \mathbf{x}_0$ is
$$E[y_0 - \hat{f}(\mathbf{x}_0)]^2 = V(\hat{f}(\mathbf{x}_0)) + [\text{Bias}(\hat{f}(\mathbf{x}_0))]^2 + V(\epsilon).$$
- Typically, as the flexibility of $\hat{f}$ increases, its bias decreases and its variance increases — a **bias-variance trade-off**.

The expected test MSE refers to the average test MSE that we would obtain if we repeatedly estimated f using a large number of different training data sets (from the same population). For notational convenience let $f(x_0) = f_0$ and $\hat{f}(x_0) = \hat{f}_0$, then

$$\begin{aligned}
E[y_0 - \hat{f}(\mathbf{x}_0)]^2 &= E[f_0 + \epsilon - \hat{f}_0]^2 \\
&= E([f_0 - \hat{f}_0]^2 + 2\epsilon(f_0 - \hat{f}_0) + \epsilon^2) \\
&= E(f_0^2 - 2f_0\hat{f}_0 + \hat{f}_0^2 + 2\epsilon f_0 - 2\epsilon\hat{f}_0 + \epsilon^2) \\
&= f_0^2 - 2f_0E(\hat{f}_0) + E(\hat{f}_0^2) + V(\epsilon) \\
&= f_0^2 - 2f_0E(\hat{f}_0) + V(\hat{f}_0) + (E(\hat{f}_0))^2 + V(\epsilon) \\
&= V(\hat{f}_0) + [E(\hat{f}_0) - f_0]^2 + V(\epsilon) \\
& \quad V(\hat{f}_0) + [\text{Bias}(\hat{f}_0)]^2 + V(\epsilon)
\end{aligned}$$

Note: $E(\hat{f}_0^2) = V(\hat{f}_0) + (E(\hat{f}_0))^2$ by definition and $2E(\epsilon\hat{f}_0) = 2E(\epsilon)E(\hat{f}_0) = 0$ because of independence.



- The bias-variance decomposition for slides 23,24 and 25. The vertical dashed line indicates the flexibility level corresponding to the smallest test mean-squared error.

The general take home message here is that as model flexibility increases, the bias tends to decrease and the variance tends to increase.

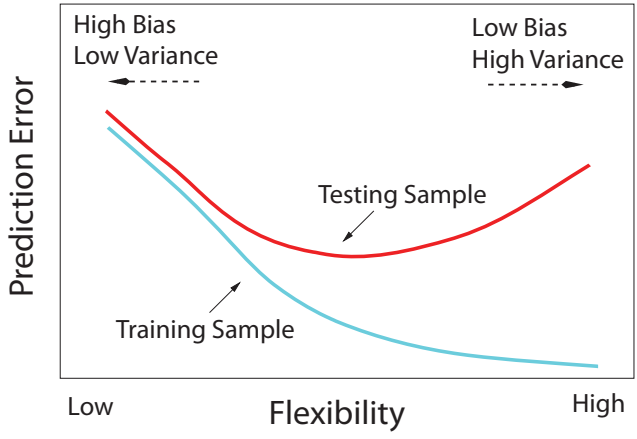
- The variance of a statistical learning method, $V(\hat{f}(x_0))$, measures how much $\hat{f}(x_0)$ is likely to change if different training data sets (from the same population) are used to fit $\hat{f}$. Flexible methods fit the training data hard and hence the model can change a lot when different training data are used

$$V(\hat{f}(x_0)) \text{ is large.}$$

- The bias, $E(\hat{f}(x_0)) - f(x_0)$, of a statistical learning method comes from trying to approximate a complicated real world problem using a relatively simple function. The bias of flexible models tends to be small because $\hat{f}(x_0) \approx y_0$ over many training data sets so that

$$E(\hat{f}(x_0)) \approx E(Y|X = x) = f(x_0).$$

Bias-variance trade-off



The figure is important and should be memorised. Our goal is to find statistical learning methods (for our particular problem) that lie near the minimum of the characteristic U shaped testing error curve.

Classification Problems

- Suppose we have a **qualitative** response Y (classification problem) that takes values from a finite (unordered) set

$$\mathcal{C} = \{y_1, y_2, \ldots, y_k\},$$

and p different predictors,

$$X = (X_1, X_2, \ldots, X_p).$$

- Our goal is to build a **classifier** $f(X)$ that assigns a **class label** from $\mathcal{C}$ to an unclassified X.

- We need to quantify uncertainty in each classification and understand how different predictors affect classifications.

Assessing Classification Accuracy

- The most common approach to assess accuracy is the **error rate** (zero-one loss function).
- Suppose we fit a classifier $\hat{f}(\mathbf{x})$ to some training data $\text{Tr} = \{\mathbf{x}_i, y_i\}_{i=1}^n$. From this set we compute the **training error rate**:

$$\text{ER}_{\text{Tr}} = \frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i),$$

  where $I(.)$ is an indicator variable that equals one if the condition is true (in this case, if $y_i \neq \hat{y}_i$) and zero otherwise.
- Suppose we have **unseen** test data $\text{Te} = \{\mathbf{x}_i, y_i\}_{i=1}^m$. From this set we compute the **testing error rate**:

$$\text{ER}_{\text{Te}} = \frac{1}{m}\sum_{i=1}^{m} I(y_i \neq \hat{y}_i).$$

Once again we are interested in classifiers with relatively low testing error rates (it's easy to fit a classifier with low (or zero in some cases) training error).

Revision material (for next slide): Let Ω denote the sample space. If $\Pr(B) > 0$, then the conditional probability of A given B is

$$\Pr(A|B) = \frac{\Pr(A \cap B)}{\Pr(B)}.$$

You can think of $\Pr(A|B)$ as the fraction of times A occurs among those in which B occurs. Another useful result is the law of total probability:

$$\Pr(B) = \sum_{i=1}^{k} \Pr(B|A_i)\Pr(A_i), \text{ where } A_1, \ldots, A_k \text{ is a partition of } \Omega \ (\cup_{i=1}^{k} A_i = \Omega \text{ and } A_j \cap A_i = \emptyset \text{ for } j \neq i).$$

Bayes Theorem: Let $A_1, \ldots, A_k$ be a partition of Ω such that $\Pr(A_i) > 0$ for each i. If $\Pr(B) > 0$, then

$$\Pr(A_i|B) = \frac{\Pr(B|A_i)\Pr(A_i)}{\Pr(B)} = \frac{\Pr(B|A_i)\Pr(A_i)}{\sum_{j=1}^{k}\Pr(B|A_j)\Pr(A_j)}$$

Bayes theorem allows us to compute $\Pr(A|B)$ if we know $\Pr(B|A)$.

Is there an ideal classifier?

- Just as in the regression setting, there can be multiple classifications at $X = \mathbf{x}$.
- The ideal $f(X)$ at $X = \mathbf{x}$ is

$$f(\mathbf{x}) = y_k \quad \textbf{if} \quad \Pr(Y = y_k | X = \mathbf{x}) = \max_{y_j \in \mathcal{C}} \Pr(Y = y_j | X = \mathbf{x})$$

  and is called **Bayes classifier**. That is, assign each observation to its most likely class given its predictor values.

- Bayes classifier is ideal in the sense that it minimizes the expected prediction error with a zero-one loss function.

Let $g_k(x) = g(x|Y = k)$ be the PDF (density) for class k and let $\pi_k = \Pr(Y = k)$ be the probability of being in class k ($\sum_i \pi_k = 1$). Then, from Bayes theorem we have that

$$\Pr(Y = k | X = \mathbf{x}) = \frac{g_k(x)\pi_k}{\sum_j g_j(x)\pi_j}.$$

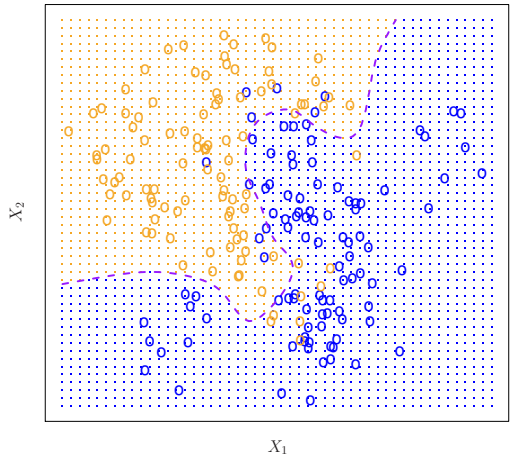
Hence, Bayes classifier is equivalent to

$$f(x) = \text{argmax}_j g_j(x)\pi_j,$$

where argmax_j means 'find the j (class) that maximizes this expression'. Hence, having $g_k(x)$ is almost equivalent to having the quantity $\Pr(Y = k|X = \mathbf{x})$. For a two-class classification problem with $Y \in \{0, 1\}$, the Bayes decision boundary is the set of points x that satisfy

$$\pi_0 g_0(x) = \pi_1 g_1(x) \quad \textbf{or} \quad \Pr(Y = 0|X = x) = \Pr(Y = 1|X = x) = 0.5.$$

That is, the set of points for which there is a 50-50 chance of being in either class. We usually don't know $g_k(x)$ or π_k so we'll have to estimate them (later in the course), but we'll go through an example on the next slide assuming we do.



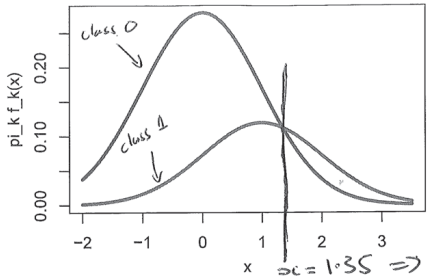
- A simulated data set consisting of two groups and the Bayes decision boundary.

If an unclassified point is in the blue region, it is classified blue. This is a two-class classification problem, so the decision boundary is the set of points x for which

$$\Pr(Y = \text{blue}|X = x) = 0.5 \qquad (\text{equivalent to } \pi_0 g_0(x) = \pi_1 g_1(x))$$

Example: Consider a two-class classification problem with $Y \in \{0, 1\}$ and $f_0(x) = \text{Normal}(0, 1)$, $f_1(x) = \text{Normal}(1, 1)$, $\pi_0 = 0.7$ and $\pi_1 = 0.3$. The Normal density function is

$$f(x) = \text{Normal}(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu)^2\right).$$



$\pi_0 f_0(x) = \pi_1 f_1(x)$

$\frac{0.7}{\sqrt{2\pi}} \exp(-\frac{1}{2}x^2) = \frac{0.3}{\sqrt{2\pi}} \exp(-\frac{1}{2}(x-1)^2)$

$2\ln(0.7/0.3) - x^2 = -x^2 + 2x - 1$

$x = \ln(0.7/0.3) + \frac{1}{2}$

Bayes boundary $\approx x = 1.35$

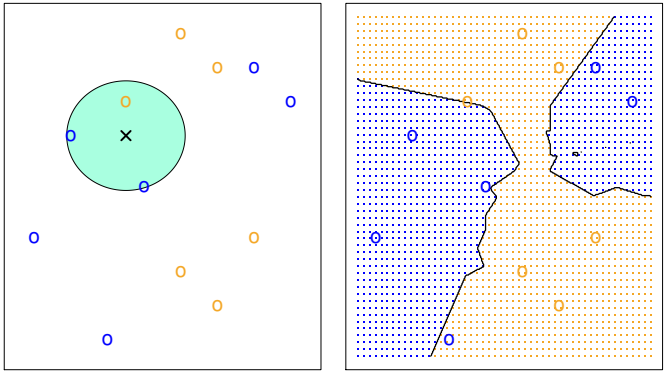
$x = 1.35 \Rightarrow$ Bayes decision boundary.

- Unfortunately, for real-world problems, we do not know the conditional distribution of Y given X so it is impossible to compute Bayes classifier.
- We can estimate the conditional distribution using nearest neighbour averaging (as we did for regression), provided we have enough local information.
- The simplest method is the k-**nearest neighbours** (KNN) classifier.

KNN Classifier

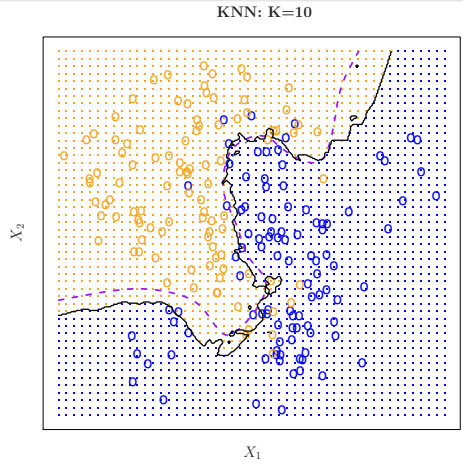
- **Goal:** Classify a test observation $\mathbf{x}_0$ to a class from $\mathcal{C} = \{y_1, y_2, \ldots, y_N\}$.
- **Steps:**
  1. Choose a positive integer k.
  2. Find the k nearest training observations to $\mathbf{x}_0$. Call this set of observations $\mathcal{N}_0$.
  3. For $j = 1, 2, \ldots, N$, estimate the probability of being in class y_j using
  $$\widehat{\Pr}(Y = y_j | X = \mathbf{x}_0) = P_j = \frac{1}{k} \sum_{i:\mathbf{x}_i \in \mathcal{N}_0} I(y_i = y_j).$$
  4. Classify $\mathbf{x}_0$ to class y_i, where $P_i \geq P_j$ for $j = 1, 2, \ldots, N$ (the majority class in $\mathcal{N}_0$).

We can measure 'closeness' using a variety of methods (more about this later in the course). For now, we can measure closeness using Euclidean distance.



- KNN decision boundary when $k = 3$ is used.

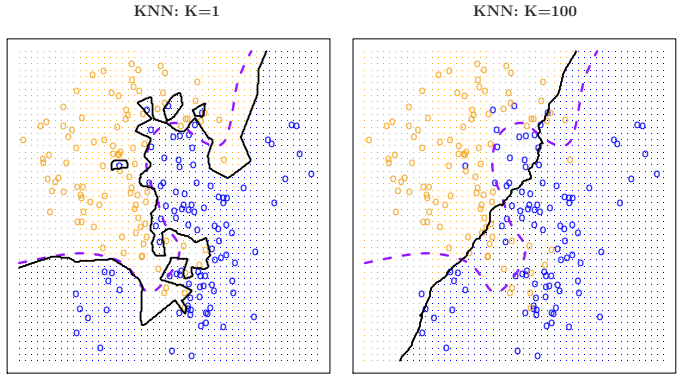
The estimated probability of x (the test observation) being blue is $2/3$, so we classify x blue. The right panel shows the *kNN* decision boundary for $k = 3$ (the set of points for which the probability of being blue (or orange) is 0.5). The shaded regions show how the predictor space is classified.



- $k = 10$ gives a very good approximation to the Bayes decision boundary.

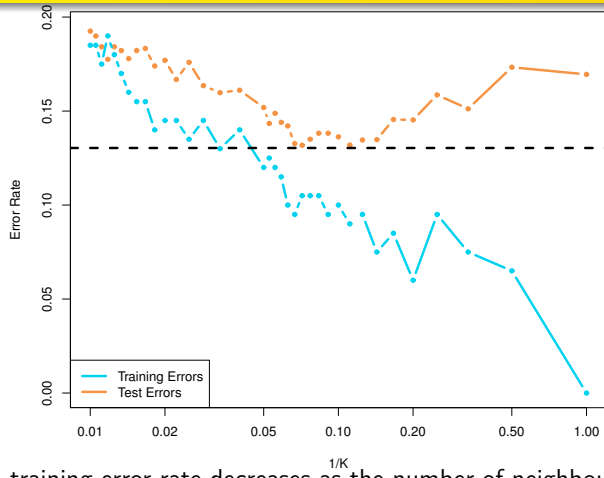
In this example *kNN* with $k = 10$ has performed extremely well. There is little difference between the Bayes decision boundary (the best we can do) and the *kNN* decision boundary.

KNN Classifier



- Voronoi Tessellation ($k = 1$) over-fit vs. $k = 100$ under-fit.

To fit a *kNN* model, the parameter k needs to be chosen by the user. We can choose the 'best k' using test data (if it's available), where we choose the k value that minimizes the testing error rate. A better approach is called cross-validation, which is covered later in the course (and in Chapter 5 of the course text).



- The KNN training error rate decreases as the number of neighbours decreases and the characteristic U shape is clear in the testing error rate.

The horizontal dashed line is Bayes error (the testing error of Bayes classifier applied to this problem, which is the best we can do). We plot $1/k$ (instead of k) so that we have increasing flexibility on the x-axis of our plot ($k = 1$ is the most flexible *kNN* method).

- If we choose k too large, we get large testing and training errors (under-fitting the training data). That is, low flexibility, high bias and low variance. Relatively large values of k may fail to capture highly non-linear decision boundaries.
- If we choose k too small, we get low training error and high testing error (over-fitting the training data). That is, high flexibility, low bias and high variance. Relatively small values of k tend to produce an overly complicated decision boundary because the boundary follows the training data too closely.

Despite its simplicity, *kNN* classification performs extremely well in a variety of settings, for example, character recognition.

Classification Problems

- Just as in the regression setting, localized classification methods start to break down as dimension increases (curse of dimensionality).

- We can make an assumption about the functional form of the decision boundary

  e.g. linearly separable;

  or make an assumption about class distributions

  e.g. normally distributed.

- We will consider a variety of parametric and non-parametric classification methods in this course.