Machine learning notes

Francesco Boi

1 Preliminary definitions

1.1 Trace of a matrix

The **trace** of a square matrix $\mathbf{A}$, denoted as $\mathbf{Tr}(\mathbf{A})$ is the sum of diagonal elements:

$$\mathbf{Tr}(\mathbf{A}) = \sum_{d=1}^{p} A_{dd} \tag{1.1}$$

It follows that $\mathrm{Tr}(\mathbf{I_d}) = p$. Also $\mathbf{Tr}(\mathbf{AB}) = \mathbf{Tr}(\mathbf{AB})$ and $\mathbf{Tr}(\mathbf{x}^T\mathbf{x}) = \mathbf{x}^T\mathbf{x}$ the latter being a scalar.

1.2 Expectation

Definition 1.1. Expectation Let X be a random variable with a finite number of outcomes $x_1, x_2, \cdots, x_k$ occurring respectively with probabilities $p_1, p_2, \cdots, p_k$. The expectation value is the summation of each outcome times its probability.

$$\mathbf{E}[X] = \sum_{k} x_k \cdot p_k \tag{1.2}$$

In case of an infinite number of outcomes the summation is replaced with the integral:

$$\mathbf{E}[X] = \int x \cdot p(x)\, dx \tag{1.3}$$

As explained here, when many random variables are involved, and there is no subscript in the E symbol, the expected value is taken with respect to their joint distribution:

$$E[h(X,Y)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(x,y) f_{XY}(x,y)\, dx\, dy$$

When a subscript is present, in some cases it tells us on which variable we should condition. So

$$\mathrm{E}_{\mathrm{X}}[\mathrm{h}(\mathrm{X},\mathrm{Y})] = \mathrm{E}[\mathrm{h}(\mathrm{X},\mathrm{Y}) \mid \mathrm{X}] = \int_{-\infty}^{\infty} \mathrm{h}(\mathrm{x},\mathrm{y}) \mathrm{f}_{\mathrm{h}(\mathrm{X},\mathrm{Y})|\mathrm{X}}(\mathrm{h}(\mathrm{x},\mathrm{y}) \mid \mathrm{x})\, \mathrm{dh}$$

...But in other cases, it tells us which density to use for the "averaging"

$$\mathrm{E}_{\mathrm{X}}[\mathrm{h}(\mathrm{X},\mathrm{Y})] = \int_{-\infty}^{\infty} \mathrm{h}(\mathrm{x},\mathrm{y}) \mathrm{f}_{\mathrm{X}}(\mathrm{x})\, \mathrm{dx}$$

1.3 Variance

Definition 1.2. Variance The variance of a random variable X is the expected value of the squared deviation from the mean of X:

$$\mathrm{Var}\,(\mathrm{X}) = \mathrm{E}\left[(\mathrm{X} - \mu)^2\right] \tag{1.4}$$

Variance can be expressed in another way recalling $\mu = \mathrm{E}\,[\mathrm{X}]$ and using the linearity property:

$$\begin{aligned} \mathrm{Var}\,(\mathrm{X}) = \mathrm{E}\left[(\mathrm{X} - \mu)^2\right] &= \mathrm{E}\left[(\mathrm{X} - \mathrm{E}\,[\mathrm{X}])^2\right] = \\ &= \mathrm{E}\left[\mathrm{X}^2 - 2 \cdot \mathrm{X} \cdot \mathrm{E}\,[\mathrm{X}] + \mathrm{E}\,[\mathrm{X}]^2\right] \\ &= \mathrm{E}\left[\mathrm{X}^2\right] - 2 \cdot \mathrm{E}\,[\mathrm{X} \cdot \mu] + \mathrm{E}\left[\mu^2\right] \\ &= \mathrm{E}\left[\mathrm{X}^2\right] - 2 \cdot \mu \cdot \mathrm{E}\,[\mathrm{X}] + \mu^2 \\ &= \mathrm{E}\left[\mathrm{X}^2\right] - \mu^2 = \mathrm{E}\left[\mathrm{X}^2\right] - \mathrm{E}\,[\mathrm{X}]^2 \end{aligned} \tag{1.5}$$

1.4 Median

Definition 1.3. Median For any probability distribution on the real line $\mathbb{R}$ with cumulative distribution function F, regardless of whether it is any kind of continuous probability distribution, in particular an absolutely continuous distribution (which has a probability density function), or a discrete probability distribution, a median is by definition any real number m that satisfies

the inequalities:

$$P(x \leq m) \geq \frac{1}{2} \quad \text{and} \quad P(x \leq m) \geq \frac{1}{2} \tag{1.6}$$

$$\tag{1.7}$$

or equivalently the inequalities

$$\int_{-\infty}^{m} F(x)dx \geq \frac{1}{2} \quad \text{and} \quad \int_{m}^{\infty} F(x)dx \geq \frac{1}{2} \tag{1.8}$$

1.5 Median as the minimizer of L_1 norm

Assume that S is a finite set, with say k elements. Line them up in order, as $s_1 < s_2 < \cdots < s_k$.

If k is even there are (depending on the exact definition of median) many medians. $|x-s_i|$ is the *distance* between x and s_i, so we are trying to minimize the sum of the distances. For example, we have k people who live at various points on the x-axis. We want to find the point(s) x such that the sum of the travel distances of the k people to x is a minimum.

Imagine that the s_i are points on the x-axis. For clarity, take $k = 7$. Start from well to the left of all the s_i, and take a tiny step, say of length ϵ, to the right. Then you have gotten ϵ closer to every one of the s_i, so the sum of the distances has decreased by 7ϵ.

Keep taking tiny steps to the right, each time getting a decrease of 7ϵ. This continues until you hit s_1. If you now take a tiny step to the right, then your distance from s_1 increases by ϵ, and your distance from each of the remaining s_i decreases by ϵ. So there is a decrease of 6ϵ, and an increase of ϵ, for a net decrease of 5ϵ in the sum.

This continues until you hit s_2. Now, when you take a tiny step to the right, your distance from each of s_1 and s_2 increases by ϵ, and your distance from each of the five others decreases by ϵ, for a net decrease of 3ϵ.

This continues until you hit s_3. The next tiny step gives an increase of 3ϵ, and a decrease of 4ϵ, for a net decrease of ϵ.

This continues until you hit s_4. The next little step brings a total increase of 4ϵ, and a total decrease of 3ϵ, for an increase of ϵ. Things get even worse when you travel further to the right. So the minimum sum of distances is reached at s_4, the median.

The situation is quite similar if k is even, say $k = 6$. As you travel to the right, there is a net decrease at every step, until you hit s_3. When you

are between s_3 and s_4, a tiny step of ϵ increases your distance from each of s_1, s_2, and s_3 by ϵ. But it decreases your distance from each of the three others, for no net gain. Thus any x in the interval from s_3 to s_4, including the endpoints, minimizes the sum of the distances.

In the even case, Some people prefer to say that any point between the two "middle" points is a median. So the conclusion is that the points that minimize the sum are the medians. Other people prefer to define the median in the even case to be the average of the two "middle" points. Then the median does minimize the sum of the distances, but some other points also do.

In formulas consider two x_i's x_1 and x_2, with $x_2 > x_1$

- $$\begin{aligned} &x_1 \leq a \leq x_2 \\ &\sum_{i=1}^{2} |x_i - a| = |x_1 - a| + |x_2 - a| = a - x_1 + x_2 - a = x_2 - x_1 \end{aligned} \tag{1.9}$$

- $$\begin{aligned} &a < x_1 \\ &\sum_{i=1}^{2} |x_i - a| = x_1 - a + x_2 - a = x_1 + x_2 - 2a \geq x_1 + x_2 - 2x_1 \\ &= x_2 - x_1 \end{aligned} \tag{1.10}$$

- $$\begin{aligned} &a \geq x_2 \\ &\sum_{i=1}^{2} |x_i - a| \quad = -x_1 + a - x_2 + a = -x_1 - x_2 + 2a \geq -x_1 - x_2 + 2x_2 = \\ &= x_2 - x_1 \end{aligned} \tag{1.11}$$

$\Longrightarrow$ for any two x_i's the sum of the absolute values of the deviations is minimum when $x_1 \leq a \leq x_2$ or $a \in [x_1, x_2]$.

When n is odd,

$$\sum_{i=1}^{n} |x_i - a| = |x_1 - a| + |x_2 - a| + \cdots + \left|x_{\frac{n-1}{2}} - a\right| + \left|x_{\frac{n+1}{2}} - a\right| + \\ + \left|x_{\frac{n+3}{2}} - a\right| + \cdots + |x_{n-1} - a| + |x_n - a| \tag{1.12}$$

consider the intervals $[x_1, x_n], [x_2, x_{n-1}], [x_3, x_{n-2}], \ldots, \left[x_{\frac{n-1}{2}}, x_{\frac{n+3}{2}}\right]$. If a is a member of all these intervals. i.e, $\left[x_{\frac{n-1}{2}}, x_{\frac{n+3}{2}}\right]$,

using the above theorem, we can say that all the terms in the sum except $\left|x_{\frac{n+1}{2}} - a\right|$ are minimized. So

$$\sum_{i=1}^{n} |x_i - a| = (x_n - x_1) + (x_{n-1} - x_2) + (x_{n-2} - x_3) + \cdots + \\ + \left(x_{\frac{n+3}{2}} - x_{\frac{n-1}{2}}\right) + \left|x_{\frac{n+1}{2}} - a\right| = \left|x_{\frac{n+1}{2}} - a\right| + \text{costant} \tag{1.13}$$

To minimize also the term $\left|x_{\frac{n+1}{2}} - a\right|$ it is clear we have to choose $a = x_{\frac{n+1}{2}}$ to get 0 but this is the definition of the median.

$\Longrightarrow$ When n is odd,the median minimizes the sum of absolute values of the deviations.

When n is even,

$$\sum_{i=1}^{n} |x_i - a| = |x_1 - a| + |x_2 - a| + \cdots + |x_{\frac{n}{2}} - a| + \\ + |x_{\frac{n}{2}+1} - a| + \cdots + |x_{n-1} - a| + |x_n - a| \tag{1.14}$$

If a is a member of all the intervals $[x_1, x_n], [x_2, x_{n-1}], [x_3, x_{n-2}], \ldots, \left[x_{\frac{n}{2}}, x_{\frac{n}{2}+1}\right]$, i.e, $a \in \left[x_{\frac{n}{2}}, x_{\frac{n}{2}+1}\right]$,

$$\sum_{i=1}^{n} |x_i - a| = (x_n - x_1) + (x_{n-1} - x_2) + (x_{n-2} - x_3) + \cdots + \\ + \left(x_{\frac{n}{2}+1} - x_{\frac{n}{2}} \right) \tag{1.15}$$

$\Longrightarrow$ When n is even, any number in the interval $[x_{\frac{n}{2}}, x_{\frac{n}{2}+1}]$, i.e, including the median, minimizes the sum of absolute values of the deviations. For example consider the series:$2, 4, 5, 10$, median, $M = 4.5$.

$$\sum_{i=1}^{4} |x_i - M| = 2.5 + 0.5 + 0.5 + 5.5 = 9$$

If you take any other value in the interval $\left[x_{\frac{n}{2}}, x_{\frac{n}{2}+1}\right] = [4, 5]$, say 4.1

$$\sum_{i=1}^{4} |x_i - 4.1| = 2.1 + 0.1 + 0.9 + 5.9 = 9$$

Taking for example 4 or 5 yields the same result:

$$\sum_{i=1}^{4} |x_i - 4| = 2 + 0 + 1 + 6 = 9$$

$$\sum_{i=1}^{4} |x_i - 5| = 3 + 1 + 0 + 5 = 9$$

This is because when summing the distance from a to the two middle points, you end up with the distance between them: $a - x_{\frac{n}{2}} + (x_{\frac{n}{2}+1} - a) = x_{\frac{n}{2}+1} - x_{\frac{n}{2}}$

For any value outside the interval $\left[x_{\frac{n}{2}}, x_{\frac{n}{2}+1}\right] = [4, 5]$, say 5.2

$$\sum_{i=1}^{4} |x_i - 5.2| = 3.2 + 1.2 + 0.2 + 4.8 = 9.4$$

1.6 Gaussian function and gaussian distribution

Definition 1.4. Gaussian function A Gaussian function is a mathematical function in the form:

$$f(x) = ae^{-\frac{(x-b)^2}{2c^2}} \tag{1.16}$$

The Gaussian function has three parameters a,b and c. The graph of a Gaussian function is the Bell curve. The parameter a is the height of the curve's peak, b is the position of the peak and c the *standard deviation* controls the width of the bell.

An important property of the Gaussian function is that the product of two Gaussian functions is still a Gaussian function:

$$\begin{aligned}
f(x)\cdot g(x) &= ae^{-\frac{(x-b)^2}{2c^2}} a_1 e^{-\frac{(x-b_1)^2}{2c_1^2}} = (a\cdot a_1)e^{-\frac{x^2-2bx+b^2+x^2-2b_1x+b_1^2}{2c^2 2c_1^2}} = \\
&= (a\cdot a_1)e^{-\frac{2x^2-2x(b+b_1)+b^2+b_1^2}{2c^2 2c_1^2}} = (a\cdot a_1)e^{-\frac{x^2-2x\frac{b+b_1}{2}+\frac{b^2+b_1^2}{2}}{2c^2c_1^2}} = \\
&= (a\cdot a_1)e^{-\frac{x^2-2x\frac{b+b_1}{2}+\frac{b^2+b_1^2}{2}}{2c^2c_1^2}} e^{\frac{\left(\frac{b+b_1}{2}\right)^2-\left(\frac{b+b_1}{2}\right)^2}{2c^2c_1^2}} = \\
&= (a\cdot a_1)e^{-\frac{x^2-2x\frac{b+b_1}{2}+\left(\frac{b+b_1}{2}\right)^2}{2c^2c_1^2}} e^{\frac{\left(\frac{b+b_1}{2}\right)^2-\frac{b^2+b_1^2}{2}}{2c^2c_1^2}} = \\
&= (a\cdot a_1)e^{-\frac{\left(x-\frac{b+b_1}{2}\right)^2}{2c^2c_1^2}} e^{\frac{\left(\frac{b+b_1}{2}\right)^2-\frac{b^2+b_1^2}{2}}{2c^2c_1^2}} = \\
&= a_2 e^{-\frac{\left(x-\frac{b+b_1}{2}\right)^2}{2c^2c_1^2}} = a_2 e^{-\frac{(x-b_2)^2}{2c_2^2}}
\end{aligned} \tag{1.17}$$

Definition 1.5. Gaussian distribution A normalized Guassian function or normal distribution is a Gaussian function with an area under the curve equal to 1. Hence it can be interpreted as a probability distribution.

To find the formula let us force the curve to have area 1:

$$\int_{-\infty}^{\infty} ae^{-\frac{(y-b)^2}{2c^2}}\,dy = a\int_{-\infty}^{\infty} e^{-\frac{(y-b)^2}{2c^2}}\,dy \tag{1.18}$$

and performing a change of integration variable:

$$\begin{aligned} &\frac{y-b}{\sqrt{2}c} = x \Rightarrow dx = dy\frac{1}{\sqrt{2}c}dx \Rightarrow dy = \sqrt{2}cdx \\ &a\int_{-\infty}^{\infty} e^{-\frac{(y-b)^2}{2c^2}}dy = \sqrt{2}ac\int_{-\infty}^{\infty} e^{-x^2}dx \end{aligned} \tag{1.19}$$

where $\sqrt{2}c$ is a constant so that in can be moved out of the integral. From now on we will focus just on the integral.

Unfortunately the integral has not solutions with elementary functions (Louiville theorem, see Abstract Algebra). However the definite integral exists (demonstration is skipped) and it is:

$$\int_{-\infty}^{\infty} e^{-x^2}dx = \sqrt{\pi} \tag{1.20}$$

To have area equal to 1 we must have:

$$ac\sqrt{2\pi} = 1 \Rightarrow a = \frac{1}{\sqrt{2\pi}c} \tag{1.21}$$

So now the function is defined by just 2 parameters.

Normally $b = \mu$ and is the mean of the function while $c = \sigma$ is the standard deviation.

As opposed to the Gaussian function, multiplying two of these functions we do not get another gaussian distribution, but a scaled guassian distribution. If the Gaussian functions have 0-mean and $\sigma^2 = 1$ then the product is still Guassian distribution with 0-mean and $\sigma^2 = 1$.

Definition 1.6. Standard Normal Gaussian distribution A Standard Normal Gaussian distribution is a Gaussian distribution with 0 mean and standard deviation 1.

2 Statistical Decision Theory

2.1 Expected prediction error

Let $X \in \mathbb{R}^p$ denote a real valued random input vector, and $Y \in \mathbb{R}^p$ a real valued random output variable, with joint distribution $Pr(X, Y)$. We seek

a function f(X) for predicting Y given values of the input X. This theory requires a loss function $L(Y, f(X))$ for penalizing errors in prediction, and by far the most common and convenient is squared error loss: $L(Y, f(X)) = (Y - f(X))^2$.

The estimated prediction error is

$$\begin{aligned} \mathrm{EPE}(f) &= \mathbf{E}\left[(Y - f(X))^2\right] = \int (y - f(x))^2 \, p(x, y) dx dy \\ &= \int_x \int_y (y - f(x))^2 \, p(x, y) dx dy \end{aligned} \tag{2.1}$$

Recalling $p(x, y) = p(y|x)p(x)$:

$$\begin{aligned} \mathrm{EPE}(f) &= \int_x \int_y (y - f(x))^2 \, p(y|x)p(x) dx dy \\ &= \int_x \int_y \left((y - f(x))^2 \, p(y|x) dy\right) p(x) dx \\ &= \int_x \mathbf{E}_{Y|X}\left[(y - f(X))^2 \, |X = x\right] p(x) dx \\ &= \mathbf{E}_X\left[\mathbf{E}_{Y|X}\left[(y - f(X))^2 \, |X = x\right]\right] \end{aligned} \tag{2.2}$$

So to minimize the prediction error:

$$f(x) = \arg\min_c \mathbf{E}_{Y|X}\left[(y - c(x))^2 \, |X = x\right] \quad \Rightarrow f(x) = \mathbf{E}\left[Y|X = x\right] \tag{2.3}$$

The *Nearest Neighbour* algorithm, assigns labels to points by counting and averaging the labels of the points belonging to a given neighbourhood:

$$\hat{f}(x) = \mathrm{Ave}\left(y_i | x_i \in N_k(x)\right) \tag{2.4}$$

where $N_k(x)$ contains the k points closest to x. This presents **two approximations**

- expectation is approximated by averaging
- conditioning at a point is relaxed to conditioning on some region centred at the target point.

With k sufficiently large, the average gets more stable and with large N the points will be more likely close to x. If $k, N \to \infty$ with $k/N \to 0$ the average becomes the expectation and we have the best possible estimator.

Unfortunately often we do not have so much data and some other times we might want exploit the supposed structure of data (linear, polynomial etc.).

However there is even a bigger problem when there are too many dimensions (i.e., p is large). Consider a uniformly distributed input in a p dimensional unit hypercube. Consider a hypercube neighbourhood around the target point capturing a fraction r of the total observations distributed among the unit hypercube. The edge of the neighbour hypercube will be $e_p(r) = r^{1/p}$. In 10 dimensions, using a neighborhood capturing 1% of the data we have $r(0.01) = 0.63$ so we must use 63% of the total data for one target.

On the contrary the linear regression is a **model-based approach**, i.e., one assumes that the function f(x) is approximately linear:

$$f(x) \approx x^T \beta \tag{2.5}$$

Putting this in the EPE equation:

$$\begin{aligned} f(x) &= \arg\min_c \mathbf{E}_{Y|X} \left[(y - c(x))^2 \, | X = x \right] \\ &= \arg\min_\beta \mathbf{E}_{Y|X} \left[\left(y - x^T \beta \right)^2 | X = x \right] \\ \Rightarrow \beta &= \left[\mathbf{E} \left[X \cdot X^T \right] \right]^{-1} \cdot \mathbf{E} \left[X \cdot Y \right] \end{aligned} \tag{2.6}$$

The minimum of a quadratic function is given by deriving and setting its derivative to 0.

If instead of a L_2 loss function we use L_1

$$\begin{aligned} \mathrm{EPE}(f) &= \mathbf{E} \left[|Y - f(X)| \right] = \int |y - f(x)| \, p(x, y) dx dy \\ &= \int_x \int_y |y - f(x)| \, p(x, y) dx dy \end{aligned} \tag{2.7}$$

Recalling $p(x, y) = p(y|x)p(x)$:

$$\begin{aligned} \mathrm{EPE}(f) &= \int_x \int_y |y - f(x)|\, p(y|x)p(x)dxdy \\ &= \int_x \int_y |(y - f(x)|\, p(y|x)dy)\, p(x)dx \\ &= \int_x \mathbf{E}_{Y|X} \left[|y - f(X)|\, |X = x\right] p(x)dx \\ &= \mathbf{E}_X \left[\mathbf{E}_{Y|X} \left[|y - f(X)|\, |X = x\right]\right] \end{aligned} \tag{2.8}$$

$$f(x) = \arg\min_c \mathbf{E}_{Y|X} \left[|y - c(x)|\, |X = x\right] \tag{2.9}$$

and as already seen in 1.5, the minimizer for the sum of distances is the median.

2.1.1 Loss function for categorical variables

For categorical output variables $\mathbb{G}_k$ we have:

$$\mathrm{EPE} = \mathbf{E}\left[L\left(G, \hat{G}(X)\right)\right] = \mathbf{E}_x \sum_{k=1}^{K} L\left[\mathbb{G}_k, \hat{G}(X)\right] \Pr\left(\mathbb{G}_k|X\right) \tag{2.10}$$

where the expectation is again taken with respect to the joint distribution $\Pr(G, X)$. Conditioning again we can write:

$$\mathrm{EPE} = \mathbf{E}_x \sum_{k=1}^{K} L\left[\mathbb{G}_k, \hat{G}(X)\right] \Pr\left(\mathbb{G}_k|X\right) \tag{2.11}$$

where the integral over y has been substituted with the summation due to the categorical nature of the variable.

The minimizer is given by:

$$\hat{G}(x) = \arg\min_{g \in \mathbb{G}} \sum_{k=1}^{K} L\left(\mathbb{G}_k, g\right) \Pr\left(\mathbb{G}_k|X = x\right) \tag{2.12}$$

Often the *zero-one loss function* is used for categorical variables and the above simplifies to:

$$\hat{G}(x) = \arg\min_{g \in \mathbb{G}} \left[1 - \Pr\left(g|X = x\right)\right] = \arg\max_{g \in \mathbb{G}} \left[\Pr\left(g|X = x\right)\right] \tag{2.13}$$

This is known as *Bayes classifier* because **it classifies to the most probable class, using conditional discrete probability distribution**.

Note: When models or loss functions use additional parameters that penalize complexity (Lasso, Ridge and others) we cannot use the training data to determine these parameters, since we would pick those that gave interpolating fits with zero residuals but it will be unlikely to predict future data.

2.2 Bias-Variance trade-off

$$\begin{aligned} \mathrm{EPE}(f) &= \mathbf{E}\left[(Y - f(X))^2\right] = \\ &= \mathbf{E}\left[Y^2\right] - 2\mathbf{E}\left[Y\right]\mathbf{E}\left[f(X)\right] + \mathbf{E}\left[f(X)^2\right] \\ &= Y^2 - 2Y\mathbf{E}\left[f(X)\right] + \mathbf{E}\left[f(X)^2\right] \end{aligned} \tag{2.14}$$

Recalling

$$\begin{aligned} &\mathrm{BIAS}\left(Y, \mathbf{E}\left[f(X)\right]\right) = \left|Y - \mathbf{E}\left[f(X)\right]\right| \\ &\Rightarrow \mathrm{BIAS}\left(Y, f(X)\right)^2 = \left(Y - \mathbf{E}\left[f(X)\right]\right)^2 \\ &= Y^2 - 2Y\mathbf{E}\left[f(X)\right] + \mathbf{E}\left[f(X)\right]^2 \end{aligned} \tag{2.15}$$

EPE can be expressed using also the Variance definition in 1.3

$$\begin{aligned} \mathrm{EPE}(f) &= Y^2 - 2Y\mathbf{E}\left[f(X)\right] + \mathbf{E}\left[f(X)^2\right] + \mathbf{E}\left[f(X)\right]^2 - \mathbf{E}\left[f(X)\right]^2 \\ &= \mathrm{BIAS}(Y, f(X)^2 + \mathrm{Var}(Y, f(x)) \end{aligned} \tag{2.16}$$

The bias is given by the distance of our predictions from real points. Complex models have more degrees of freedom and are able to fit closer real points hence they to tend to have low bias. However, they present higher variance. On the contrary simple models (i.e., linear) have lower variance but higher bias.

The error due to variance is the amount by which the prediction, over one training set, differs from the expected predicted value, over all the training sets. As with bias, you can repeat the entire model building process multiple times. To paraphrase Manning et al (2008), variance measures how inconsistent are the predictions from one another, over different training sets, not whether they are accurate or not. A learning algorithm with low bias must be "flexible" so that it can fit the data well. But if the learning algorithm is too flexible, it will fit each training data set differently, and hence have high variance.

For linear models fit by ordinary least squares, the estimation bias is zero. For restricted fits, such as ridge regression, it is positive, and we trade it off with the benefits of a reduced variance.

3 Linear Regression Models

We start from the *univariate linear regression*, i.e., each output consists of a single value while the input is a vector of values.

3.1 Univariate linear regression

Univariate means single output, i.e, y is a number. The basic form is

$$\mathrm{f(X)} = \beta_0 + \sum_{\mathrm{j–1}}^{\mathrm{p}} \mathrm{X_j}\beta_\mathrm{j} \tag{3.1}$$

The most popular estimation method for a linear model is the least square:

$$\mathrm{RSS}(\beta) = \sum_{\mathrm{i=1}}^{\mathrm{p}} (\mathrm{y_i} - \mathrm{f(x_i)})^2 = (\mathbf{y} - \mathbf{X}\beta)^{\mathrm{T}} (\mathbf{y} - \mathbf{X}\beta) \tag{3.2}$$

where $\mathbf{X}$ is a $\mathrm{N} \times (\mathrm{p} + 1)$ matrix with each row being an input vector, $\mathbf{y}$ a N vector (we must have N input-output pairs, in this case the output is considered mono-dimensional).

By minimizing we get:

$$\hat{\beta} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1} \mathbf{X}^{\mathrm{T}}\mathrm{Y} \tag{3.3}$$

Geometrically we are projecting y onto the hyperplane spanned by X and the projections is referred to as $\hat{y}$:

$$\hat{y} = \mathbf{X}\left(\mathbf{X}^{T}\mathbf{X}\right)^{-1}\mathbf{X}^{T}Y = HY \tag{3.4}$$

where H is called the *hat* matrix.

3.2 Equivalence of Ordinary least squares and maximum likelihood

We are using an additive model, assuming a Gaussian white noise:

$$y = \beta^{T}\mathbf{x} + \epsilon \tag{3.5}$$

$$\epsilon \sim \mathcal{N}(0, \sigma^2) \tag{3.6}$$

Adding a constant to a Gaussian random variable is equivalent to another Gaussian random variable with the mean shifted:

$$\Pr(y) \sim \mathcal{N}(\beta^{T}\mathbf{x}, \sigma^2) \tag{3.7}$$

Considering the matrix $\mathbf{X}$ and the output vector $\mathbf{y}$ representing the training set, used to estimate the coefficients, we have:

$$\Pr(\mathbf{y}|\mathbf{X}, \beta, \sigma^2) = \prod_{i=1}^{N} \Pr(y_i|\mathbf{x}, \beta, \sigma^2) = \prod_{i=1}^{N} \mathcal{N}(\beta^{T}\mathbf{x}, \sigma^2) \tag{3.8}$$

where we have assumed each observation is independent. A product of univariate Gaussian can be rewritten as a multivariate Gaussian:

$$\begin{aligned}
\Pr(\mathbf{y}|\mathbf{X}, \beta, \sigma^2) &= \prod_{i=1}^{N} \mathcal{N}(\beta^{T}\mathbf{x}, \sigma^2) = \prod_{i=1}^{N} \frac{1}{(2\pi)^{\frac{1}{2}}\sigma} e^{-\frac{\left(y_i - \beta^{T}\mathbf{x}_i\right)^2}{2\sigma^2}} = \\
&= \frac{1}{(2\pi)^{\frac{p}{2}}\sigma} \prod_{i=1}^{N} e^{-\sum_{i=1}^{N} \frac{\left(y_i - \beta^{T}\mathbf{x}_i\right)^2}{2\sigma^2}} = \\
&= \frac{1}{(2\pi)^{\frac{p}{2}}\sigma|\mathbf{I}|} e^{-\frac{1}{2}(\mathbf{y} - \beta^{T}\mathbf{X})^{T}(\sigma^2\mathbf{I})^{-1}(\mathbf{y} - \beta^{T}\mathbf{X})} = \mathcal{N}(\beta^{T}\mathbf{X}, \sigma^2\mathbf{I})
\end{aligned} \tag{3.9}$$

If the variables are not independent the more general form is:

$$\mathcal{N}(\beta^{T}\mathbf{X}, \mathbf{\Sigma}) = \frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{y} - \beta^{T}\mathbf{X})^{T}\Sigma^{-1}(\mathbf{y} - \beta^{T}\mathbf{X})} \tag{3.10}$$

Definition of likelihood The quantity $\Pr(\mathrm{y}|\mathbf{x}, \beta, \sigma^2)$ is called **likelihood** and tell us how much it is likely the outcome $\mathrm{y_i}$ in the dataset given the input $\mathbf{x}$ and the parameters.

A different approach to find a model that fits the data is to maximize the *likelihood* of the whole dataset:

$$\mathrm{L} = \Pr(\mathbf{y}|\mathbf{X}, \beta, \sigma^2) = \frac{1}{(2\pi)^{\frac{\mathrm{p}}{2}}|\Sigma|^{\frac{1}{2}}} \mathrm{e}^{-\frac{1}{2}(\mathbf{y}-\beta^{\mathrm{T}}\mathbf{X})^{\mathrm{T}}\Sigma^{-1}(\mathbf{y}-\beta^{\mathrm{T}}\mathbf{X})} \tag{3.11}$$

Actually maximizing the likelihood is equivalent to maximizing its logarithmic, the *log-likelihood*:

$$\begin{aligned}
\log \mathrm{L} &= \sum_{\mathrm{i}=1}^{\mathrm{N}} \log \left[\frac{1}{(2\pi)^{\frac{1}{2}}\sigma} \mathrm{e}^{-\frac{\left(\mathrm{y_i}-\beta^{\mathrm{t}}\mathbf{x_i}\right)^2}{2\sigma^2}} \right] \\
&= \sum_{\mathrm{i}=1}^{\mathrm{N}} -\frac{1}{2}\log 2\pi - \log\sigma - \frac{\left(\mathrm{y_i} - \beta^{\mathrm{T}}\mathbf{x_i}\right)^2}{2\sigma^2} = \\
&= -\frac{1}{2}\log 2\pi - \log\sigma - \frac{1}{2\sigma^2}\sum_{\mathrm{i}=1}^{\mathrm{N}} \left(\mathrm{y_i} - \beta^{\mathrm{T}}\mathbf{x_i}\right)^2
\end{aligned} \tag{3.12}$$

As already done for OLS, taking the derivative w.r.t. β and setting it to 0:

$$\begin{aligned}
&\frac{\partial \log \mathrm{L}}{\partial \beta} = -\frac{1}{2\sigma^2}(-2)\left(\mathbf{y_i} - \beta^{\mathrm{T}}\mathbf{x_i}\right) = 0 \\
&\Rightarrow \left(\mathbf{y} - \beta^{\mathrm{T}}\mathbf{X}\right) = 0 \Rightarrow \beta = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}
\end{aligned} \tag{3.13}$$

This is the same solution of the OLS: the two models are equivalent.

The two models are equivalent assuming a normal distribution.

3.3 Expectation of the parameter estimation: unbiased estimator

Computing the expectation of $\hat{\beta}$:

$$\begin{aligned}
&\mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)}\left[\hat{\beta}\right] = \sum \hat{\beta}\Pr(\mathbf{y}|\mathrm{X},\beta,\sigma^2) = \\
&= \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\sum \mathbf{y}\Pr(\mathbf{y}|\mathrm{X},\beta,\sigma^2) = \\
&= \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)}\left[\mathbf{y}\right] = \\
&= \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{X}\beta = \beta
\end{aligned} \tag{3.14}$$

This is an **unbiased estimator**.
Now let us calculate the covariance matrix:

$$\begin{aligned}
&\mathrm{Cov}\left[\hat{\beta}\right] = \\
&= \mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)}\left[\left(\hat{\beta} - \mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)}\left[\beta\right]\right)\left(\hat{\beta} - \mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)}\left[\beta\right]\right)^{\mathrm{T}}\right] = \\
&= \mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)}\left[\left(\hat{\beta} - \beta\right)\left(\hat{\beta} - \beta\right)^{\mathrm{T}}\right] = \\
&= \mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)}\left[\hat{\beta}\hat{\beta}^{\mathrm{T}}\right] - \beta\beta^{\mathrm{T}}
\end{aligned} \tag{3.15}$$

and

$$\begin{aligned}
&\mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)}\left[\hat{\beta}\hat{\beta}^{\mathrm{T}}\right] = \\
&= \mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)}\left[\left(\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}\right)\left(\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}\right)^{\mathrm{T}}\right] = \\
&= \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)}\left[\mathbf{y}\mathbf{y}^{\mathrm{T}}\right]\mathbf{X}\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}
\end{aligned} \tag{3.16}$$

and recalling from 3.9 $\Pr(\mathbf{y}) \sim \mathcal{N}(\beta^{\mathrm{T}}\mathbf{X}, \sigma^2\mathbf{I})$

$$\begin{aligned}
&\mathrm{Cov}\left[\mathbf{y}\right] = \sigma^2\mathbf{I} = \\
&= \mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)}\left[\mathbf{y}\mathbf{y}^{\mathrm{T}}\right] - \mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)}\left[\mathbf{y}\right]\mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)}\left[\mathbf{y}^{\mathrm{T}}\right]
\end{aligned} \tag{3.17}$$

Rearranging

$$\begin{aligned}
&\Rightarrow \mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)}\left[\mathbf{y}\mathbf{y}^{\mathrm{T}}\right] = \sigma^2\mathbf{I} + \mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)}\left[\mathbf{y}\right]\mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)}\left[\mathbf{y}^{\mathrm{T}}\right] = \\
&= \sigma^2\mathbf{I} + \mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)}\left[\mathbf{X}\beta\right]\mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)}\left[\beta^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\right] = \\
&= \sigma^2\mathbf{I} + \mathbf{X}\beta\beta^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}
\end{aligned} \tag{3.18}$$

Substituting:

$$\begin{aligned}
&\mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)}\left[\hat{\beta}\hat{\beta}^{\mathrm{T}}\right] = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\left(\sigma^2\mathbf{I} + \mathbf{X}\beta\beta^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\right)\mathrm{X}\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1} = \\
&= \sigma^2\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}
\end{aligned} \tag{3.19}$$

and finally variance-covariance matrix of the least square parameters is

$$\mathrm{Cov}\left[\hat{\beta}\right] = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\sigma^2 \tag{3.20}$$

$$\mathrm{Var}(\hat{\beta}) = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\sigma^2 \tag{3.21}$$

3.4 Noise variance estimation

We can find an estimation of the noise variance from the maximum likelihood model using the same procedure used to find the parameters, i.e., taking the derivative and equating it to 0:

$$\begin{aligned}
&\frac{\partial \log \mathrm{L}}{\partial\sigma} = \sum_{\mathrm{i}=1}^{\mathrm{N}} -\frac{1}{\sigma} + \frac{1}{\sigma^3}\left(\mathrm{y_i} - \mathbf{x}_{\mathrm{i}}^{\mathrm{T}}\beta\right)^2 = 0 \Rightarrow \frac{\mathrm{N}}{\sigma} + \frac{1}{\sigma^3}\sum_{\mathrm{i}=1}^{\mathrm{N}}\left(\mathrm{y_i} - \mathbf{x}^{\mathrm{T}}\beta\right)^2 = 0 \\
&\Rightarrow \hat{\sigma^2} = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}}\left(\mathrm{y_i} - \mathbf{x}_{\mathrm{i}}^{\mathrm{T}}\beta\right)^2
\end{aligned} \tag{3.22}$$

This can be re-expressed as

$$
\begin{aligned}
&\Rightarrow \hat{\sigma^2} = \frac{1}{\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \left(\mathrm{y_i} - \mathbf{x}^{\mathrm{T}} \hat{\beta} \right)^2 = \frac{1}{\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \left(\mathrm{y_i} - \mathbf{x}_{\mathrm{i}}^{\mathrm{T}} \left(\mathbf{X}^{\mathrm{T}} \mathbf{X} \right)^{-1} \mathbf{X}^{\mathrm{T}} \mathbf{y} \right)^2 = \\
&= \frac{1}{\mathrm{N}} \left(\mathbf{y} - \mathbf{X} \left(\mathbf{X}^{\mathrm{T}} \mathbf{X} \right)^{-1} \mathbf{X}^{\mathrm{T}} \mathbf{y} \right)^{\mathrm{T}} \left(\mathbf{y} - \mathbf{X} \left(\mathbf{X}^{\mathrm{T}} \mathbf{X} \right)^{-1} \mathbf{X}^{\mathrm{T}} \mathbf{y} \right) = \\
&= \mathbf{y}^{\mathrm{T}} \mathbf{y} - 2 \mathbf{y}^{\mathrm{T}} \mathbf{X} \left(\mathbf{X}^{\mathrm{T}} \mathbf{X} \right)^{-1} \mathbf{X}^{\mathrm{T}} \mathbf{y} + \mathbf{y}^{\mathrm{T}} \cancel{\mathbf{X} \left(\mathbf{X}^{\mathrm{T}} \mathbf{X} \right)^{-1}} \cancel{\mathbf{X}^{\mathrm{T}} \mathbf{X}} \left(\mathbf{X}^{\mathrm{T}} \mathbf{X} \right)^{-1} \mathbf{X}^{\mathrm{T}} \mathbf{y} = \\
&= \mathbf{y}^{\mathrm{T}} \mathbf{y} - \mathbf{y}^{\mathrm{T}} \mathbf{X} \left(\mathbf{X}^{\mathrm{T}} \mathbf{X} \right)^{-1} \mathbf{X}^{\mathrm{T}} \mathbf{y} = \left(\mathbf{y}^{\mathrm{T}} \mathbf{y} - \mathbf{y}^{\mathrm{T}} \mathbf{X} \hat{\beta} \right)
\end{aligned}
\tag{3.23}
$$

Taking the expectation w.r.t. $\Pr(\mathbf{y}|\mathbf{X}, \beta, \sigma^2)$:

$$
\begin{aligned}
&\mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)} \left[\hat{\sigma}^2 \right] = \frac{1}{\mathrm{N}} \mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)} \left[\mathbf{y}^{\mathrm{T}} \mathbf{y} - \mathbf{y}^{\mathrm{T}} \mathbf{X} \hat{\beta} \right] = \\
&= \frac{1}{\mathrm{N}} \mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)} \left[\mathbf{y}^{\mathrm{T}} \mathbf{y} \right] - \frac{1}{\mathrm{N}} \mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)} \left[\mathbf{y}^{\mathrm{T}} \mathbf{X} \left(\mathbf{X}^{\mathrm{T}} \mathbf{X} \right)^{-1} \mathbf{X}^{\mathrm{T}} \mathbf{y} \right]
\end{aligned}
\tag{3.24}
$$

Suppose $\mathbf{t} \sim \mathcal{N}(\mu, \mathbf{\Sigma})$, then $\mathbf{E}_{\mathrm{p}(\mathbf{t})} \left(\mathbf{t}^{\mathbf{T}} \mathbf{A} \mathbf{t} \right) = \mathbf{Tr}(\mathbf{A}\mathbf{\Sigma}) + \mu^{\mathbf{T}} \mathbf{A} \mu$ with $\mu = \mathbf{X}\beta$

$$
\begin{aligned}
&\mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)}\left[\hat{\sigma}^2\right] = \frac{1}{\mathrm{N}}\mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)}\left[\mathbf{y}^{\mathrm{T}}\mathbf{y}\right] + \\
&-\frac{1}{\mathrm{N}}\mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)}\left[\mathbf{y}^{\mathrm{T}}\mathbf{X}\Big(\mathbf{X}^{\mathrm{T}}\mathbf{X}\Big)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}\right] = \\
&= \frac{1}{\mathrm{N}}\mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)}\left[\mathbf{y}^{\mathrm{T}}\mathbf{I}_{\mathrm{N}}\mathbf{y}\right] + \\
&-\frac{1}{\mathrm{N}}\mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)}\left[\mathbf{y}^{\mathrm{T}}\mathbf{X}\Big(\mathbf{X}^{\mathrm{T}}\mathbf{X}\Big)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}\right] = \\
&= \frac{1}{\mathrm{N}}\left(\mathbf{Tr}\left(\sigma^2\mathbf{I}_{\mathrm{N}}\right) + \beta^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\beta\right) + \\
&-\frac{1}{\mathrm{N}}\left(\mathbf{Tr}\left[\sigma^2\mathbf{X}\Big(\mathbf{X}^{\mathrm{T}}\mathbf{X}\Big)^{-1}\mathbf{X}^{\mathrm{T}}\right] + \beta^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\Big(\mathbf{X}^{\mathrm{T}}\mathbf{X}\Big)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{X}\beta\right) = \\
&= \frac{1}{\mathrm{N}}\left(\mathrm{N}\sigma^2 + \beta^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\beta\right) - \frac{1}{\mathrm{N}}\left(\sigma^2\mathbf{Tr}\left[\mathbf{X}\Big(\mathbf{X}^{\mathrm{T}}\mathbf{X}\Big)^{-1}\mathbf{X}^{\mathrm{T}}\right] + \beta^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\beta\right) = \\
&= \sigma^2 + \cancel{\frac{1}{\mathrm{N}}\beta^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\beta} - \frac{\sigma^2}{\mathrm{N}}\mathbf{Tr}\left[\mathbf{X}\Big(\mathbf{X}^{\mathrm{T}}\mathbf{X}\Big)^{-1}\mathbf{X}^{\mathrm{T}}\right] \cancel{-\frac{1}{\mathrm{N}}\beta^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathrm{X}\beta} = \\
&= \sigma^2 - \frac{\sigma^2}{\mathrm{N}}\mathbf{Tr}\left[\Big(\mathbf{X}^{\mathrm{T}}\mathbf{X}\Big)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{X}\right] = \sigma^2 - \frac{\sigma^2}{\mathrm{N}}\mathbf{Tr}\left[\mathbf{I}_{\mathrm{p}}\right] \\
&\Rightarrow \mathbf{E}_{\Pr(\mathbf{y}|\mathbf{X},\beta,\sigma^2)}\left[\hat{\sigma}^2\right] = \sigma^2\left(1 - \frac{\mathrm{p}}{\mathrm{N}}\right)
\end{aligned}
\tag{3.25}
$$

where lastly we have used the product property of the trace (see 1.1).

Normally $\mathrm{p} < \mathrm{N}$ hence the estimate of the variance is smaller than the true variance, so this estimator is **biased**. The estimate gets closer to the real value when p/N is small, i.e., assuming p is fixed, increasing the samples used.

This result might be strange. First of all notice from 3.22 that the closer the model gets to the data, the smaller $\hat{\sigma}^2$. By definition the parameter estimates are the ones that minimise the noise and hence $\hat{\sigma}^2$. As a consequence, when using the true parameters we would get equal or higher variance.

To estimate the true variance we can use the following formula:

$$
\hat{\sigma}^2 = \frac{1}{\mathrm{N} - \mathrm{p} - 1}\sum_{\mathrm{i}=1}^{\mathrm{N}}(\mathrm{y_i} - \hat{\mathrm{y}}_\mathrm{i})^2 \tag{3.26}
$$

$\mathrm{N} - \mathrm{p} - 1$ makes this estimation unbiased i.e., $\frac{1}{\mathrm{N}-\mathrm{p}-1}\mathbf{E}[\hat{\sigma}^2] = \sigma^2$.

3.5 Interpretation of covariance

Consider a covariance matrix of size 2×2 for a two parameter model (i.e., a line on the plane) and suppose the first diagonal element, corresponding to the variable $\hat{\beta}_0$ is much bigger than the second one corresponding to $\hat{\beta}_1$. This means that we can change $\hat{\beta}_0$ a little without affecting too much the model. On the contrary if the variance is small, small changes will affect significantly the model. Sometimes this happens when one variable has a much higher absolute value.

If the values on the off-diagonals are negative, then when increasing one coefficient i.e., $\hat{\beta}_0$, the other must be decreased to have the line to pass as close as possible to all points. For example in 2D, increasing $\hat{\beta}_0$ reduces the coefficient value: if it is positive, the line will be "more horizontal", if negative it becomes steeper, "more vertical".

3.6 Z-score

Let us assume data were really generated by a linear model but were corrupted by Gaussian noise with 0 mean and variance σ^2:

$$\mathrm{Y} = \beta_0 + \sum_{\mathrm{i}}^{\mathrm{p}} \beta_{\mathrm{i}} \mathrm{X}_{\mathrm{i}} + \epsilon \tag{3.27}$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. The estimated parameters will still be a normal distribution:

$$\hat{\beta} \sim \mathcal{N}(\beta, (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\sigma^2) \tag{3.28}$$

Definition 3.1. Z-score A Z-score is a numerical measurement used in statistics of a value's relationship to the mean (average) of a group of values, measured in terms of standard deviations from the mean. If a Z-score is 0, it indicates that the data point's score is identical to the mean score. A Z-score of 1.0 would indicate a value that is one standard deviation from the mean. Z-scores may be positive or negative, with a positive value indicating the score is above the mean and a negative score indicating it is below the mean. Z-scores are measures of an observation's variability

$$z_j = \frac{x - \mu}{\sigma} \tag{3.29}$$

In machine learning the z-value is the regression coefficient divided by its standard error. It is also sometimes called the z-statistic. If the z-value is too big in magnitude (i.e., either too positive or too negative), it indicates that the corresponding true regression coefficient is not 0 and the corresponding X-variable matters. A good rule of thumb is to use a cut-off value of 2 which approximately corresponds to a two-sided hypothesis test with a significance level of $\alpha = 0.05$.

Z-values are computed as the test statistic for the hypothesis test that the true corresponding regression coefficient β is 0. In hypothesis testing, we assume the null hypothesis is true, and then see if the data provide evidence against it. So in this case, we assume β is 0. That is, we assume the expectation of the fitted regression coefficient $\hat{\beta}$ is 0:

$$z_j = \frac{\hat{\beta}_j}{\hat{\sigma}\sqrt{v_j}} \tag{3.30}$$

where the denominator is the variance of the parameter (from 3.28) with v_j being the diagonal element of $(X^T X)^{-1}$.

Theorem 1 (**The Gauss-Markov theorem**)**.** Among all linear unbiased estimators, the least square estimates of the parameters are the ones having smallest variance and consequently from 2.16 is the one with the smallest mean squared error (the bias-squared term for unbiased estimator is by definition 0).

Proof. Let $\hat{\beta} = Cy$ be another linear estimator of β with $C = (X^T X)^{-1} X^T + D$

$$\mathbf{E}\left[\hat{\beta}\right] = \mathbf{E}\left[Cy\right] = \mathbf{E}\left[\left(\left(X^T X\right)^{-1} X^T + D\right)(X\beta + \epsilon)\right] = \tag{3.31}$$

$$= \left(\left(X^T X\right)^{-1} X^T X\beta + DX\beta\right) + \cancel{\left(\left(X^T X\right)^{-1} X^T + D\right)\mathbf{E}[\epsilon]} = \tag{3.32}$$

$$= (\beta + DX\beta) = (I + DX)\,\beta \tag{3.33}$$

$$\tag{3.34}$$

where $\mathbf{E}[\epsilon] = 0$.

To be an unbiased estimator $\mathrm{DX} = 0$, then

$$
\begin{aligned}
\mathrm{Var}(\hat{\beta}) &= \mathrm{Var}(\mathrm{Cy}) = \mathrm{C}\mathrm{Var}(\mathrm{y})\mathrm{C}^{\mathrm{T}} = \sigma^2 \mathrm{C}\mathrm{C}^{\mathrm{T}} = \\
&= \sigma^2 \left(\left(\mathrm{X}^{\mathrm{T}}\mathrm{X} \right)^{-1} \mathrm{X}^{\mathrm{T}} + \mathrm{D} \right) \left(\left(\mathrm{X}^{\mathrm{T}}\mathrm{X} \right)^{-1} \mathrm{X}^{\mathrm{T}} + \mathrm{D} \right)^{\mathrm{T}} = \\
&= \sigma^2 \left(\left(\mathrm{X}^{\mathrm{T}}\mathrm{X} \right)^{-1} \mathrm{X}^{\mathrm{T}} + \mathrm{D} \right) \left(\mathrm{X} \left(\mathrm{X}^{\mathrm{T}}\mathrm{X} \right)^{-1} + \mathrm{D}^{\mathrm{T}} \right) = \\
&= \sigma^2 (\cancel{\left(\mathrm{X}^{\mathrm{T}}\mathrm{X} \right)^{-1} \mathrm{X}^{\mathrm{T}}\mathrm{X}} \left(\mathrm{X}^{\mathrm{T}}\mathrm{X} \right)^{-1} + \left(\mathrm{X}^{\mathrm{T}}\mathrm{X} \right)^{-1} \mathrm{X}^{\mathrm{T}}\mathrm{D}^{\mathrm{T}} + \\
&\quad + \mathrm{DX} \left(\mathrm{X}^{\mathrm{T}}\mathrm{X} \right)^{-1} + \mathrm{D}\mathrm{D}^{\mathrm{T}}) = \\
&= \sigma^2 (\cancel{\left(\mathrm{X}^{\mathrm{T}}\mathrm{X} \right)^{-1} \mathrm{X}^{\mathrm{T}}\mathrm{X}} \left(\mathrm{X}^{\mathrm{T}}\mathrm{X} \right)^{-1} + \left(\mathrm{X}^{\mathrm{T}}\mathrm{X} \right)^{-1} (\mathrm{DX})^{\mathrm{T}} + \\
&\quad + \mathrm{DX} \left(\mathrm{X}^{\mathrm{T}}\mathrm{X} \right)^{-1} + \mathrm{D}\mathrm{D}^{\mathrm{T}}) \\
&\mathrm{DX} = 0 \\
\Rightarrow \mathrm{Var}(\tilde{\beta}) &= \sigma^2 \left(\left(\mathrm{X}^{\mathrm{T}}\mathrm{X} \right)^{-1} + \cancel{\left(\mathrm{X}^{\mathrm{T}}\mathrm{X} \right)^{-1} (\mathrm{DX})^{\mathrm{T}}} + \cancel{\mathrm{DX} \left(\mathrm{X}^{\mathrm{T}}\mathrm{X} \right)^{-1}} + \mathrm{D}\mathrm{D}^{\mathrm{T}} \right) \\
\mathrm{Var}(\tilde{\beta}) &= \mathrm{Var}(\hat{\beta}) + \sigma^2 \mathrm{D}\mathrm{D}^{\mathrm{T}}
\end{aligned}
\tag{3.35}
$$

□

3.7 Orthogonalization

Normally inputs are not perpendicular but can be orthogonalized. The goal is to define a new orthogonal basis for the data. The procedure, named **Grand-Schmidt procedure** is the following:

When inputs are correlated, the residual will be close to zero generating instabilities in the coefficients $\hat{\beta}_{\mathrm{j}}$ and the z-score will be small.

The algorithm in matrix form is

$$\mathbf{X} = \mathbf{Z}\boldsymbol{\Gamma}$$

where $\mathrm{z_j}$ are the column vectors of Z and Γ is upper triangular. Introducing the diagonal matrix D with j-th diagonal element $\mathrm{D_{jj}} = ||\mathrm{z_j}||$ we get:

initialize $z_0 = x_0 = \mathbf{1}$ where $\mathbf{1}$ is a vector of all ones;
For $j = 1, \cdots, p$ regress x_j on $z_0, \cdots, z_{p-1}$ to produce the coefficients $\hat{\gamma}_{lj} = \frac{\langle z_l, x_j \rangle}{\langle z_l, z_l \rangle}$ for $l = 0, \cdots, j-1$;
Calculate the residual vectors as $z_j = x_j - \sum_{k=0}^{j-1} \hat{\gamma}_{kj} z_k$;
Regress $\mathbf{y}$ on the residual z_p to get the estimate $\hat{\beta}_p = \frac{\langle z_p, y \rangle}{\langle z_p, z_p \rangle}$

$$\mathbf{X} = \mathbf{Z}\mathbf{D}^{-1}\mathbf{D}\boldsymbol{\Gamma} = \mathbf{Q}\mathbf{R} \tag{3.36}$$

where Q is an orthogonal, $Q^T Q = I$, $N \times (p+1)$ matrix and R is a $(p+1)(p+1)$ upper triangular matrix.

The least square solution becomes

$$\hat{\beta} = R^{-1} Q^T y \hat{y} = Q Q^T y \tag{3.37}$$

3.8 Multivariate output

We can rewrite the equation in matrix form:

$$Y = XB + E \tag{3.38}$$

where Y is a $N \times K$ matrix, X is $N \times (p+1)$, B is $(p+1) \times K$, E has the same dimensions of Y. The root squared error is

$$\begin{aligned} RSS(B) &= \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f(x_i))^2 = \mathrm{tr}\left[(Y - XB)^T (Y - XB) \right] \\ \Rightarrow B &= \left(X^T X \right)^{-1} X^T Y \end{aligned} \tag{3.39}$$

Multiple outputs do not affect one another's least square estimates.

3.9 Subset selection

Least square estimates sometimes show low bias but high variance resulting in a non-satisfactory prediction accuracy. This can be improved by setting some coefficients to 0 to sacrifice a little of bias to reduce significantly the variance.

Other times it is useful to reduce the input dimensionality for easiness of interpretation or for computation purposes.

3.9.1 Best subset selection

This algorithm finds for each k $\in 0, 1, 2, \cdots$ p, the subset of size k that gives the smallest residual sum of squares. Note that if a variable is in the best subset of size m, it might not be in the subsets of larger size (and of course neither in the smallest ones).

3.9.2 Forward stepwise selection

Searching for all the subsets is too computationally intensive (and infeasible for p > 40). The *forward stepwise* algorithm starts with the intercept and then sequentially adds to the model the predictor that most improves the fit. QR decomposition can be exploited to choose the next candidate.

This algorithm is a greedy sub-optimal algorithm. Statistically it will have lower variance but higher bias.

3.9.3 Backward stepwise selection

Backward stepwise selection starts with the full model and sequentially removes the predictor having least impact on the model, i.e., having the smallest Z – score.

Note: this algorithm can only be applied when N $>$ p while *forward stepwise* can always be used.

3.9.4 Implementations

[From ESLII pg. 60] Some software packages implement hybrid stepwise-selection strategies that consider both forward and backward moves at each step, and select the "best" of the two. For example in the R package the step function uses the AIC criterion for weighing the choices, which takes proper account of the number of parameters fit; at each step an add or drop will be performed that minimizes the AIC score.

Other more traditional packages base the selection on F-statistics, adding "significant" terms, and dropping "non-significant" terms. These are out of fashion.

3.9.5 Forward stagewise regression

As forward stepwise, it starts with the intercept. At each step the algorithm identifies the variable most correlated with the residual and computes the linear regression coefficient of the residual on this chosen variable and then adds it to the current coefficient for that variable. This is continued till none of the variables have correlation with the residual.

At each step only one coefficient is updated by a small step so that the number of steps is bigger than p. This slow-fitting pays in high dimensions.

Note: forward stagewise is very competitive in

3.10 Shrinkage methods

Subset selection methods either keep or remove a predictor. It has higher variance. Shrinkage or regularization methods are more continuous. They force the model to keep the weights as small as possible.

3.10.1 Ridge regression

Ridge regression shrinks the coefficients by imposing a penalty on their size. The ridge coefficients minimize a penalized residual sum of squared errors

$$\hat{\beta}^{\text{ridge}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\} \tag{3.40}$$

where λ is a parameter controlling the amount of shrinkage: the bigger the value the greater the amount of shrinkage. This concept is also used in the Neural Networks. Another way to express 3.40 is the following:

$$\begin{aligned} \hat{\beta}^{\text{ridge}} &= \arg\min_{\beta} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \right\} \\ &\text{subject to} \sum_{j=1}^{p} \beta_j^2 \leq t \end{aligned} \tag{3.41}$$

where there is a one-to-one correspondence between λ and t. Note that the penalization term does not consider β_0 otherwise the procedure will depend on the chosen origin for Y.

This algorithm solves the problem of high variance in case of correlated inputs when big coefficients of correlated variables can be cancelled out. With a constraints on the coefficients this problem is alleviated.

The coefficients are not preserved when the input is scaled. Generally inputs are standardized before applying the algorithm.

$$\mathrm{RSS}(\lambda, \beta) = (\mathrm{y} - \mathrm{X}\beta)^{\mathrm{T}} (\mathrm{y} - \mathrm{X}\beta) + \lambda\beta^{\mathrm{T}}\beta \tag{3.42}$$

$$\hat{\beta}^{\mathrm{ridge}} = \left(\mathrm{X}^{\mathrm{T}}\mathrm{X} + \lambda\mathrm{I}\right)^{-1} \mathrm{X}^{\mathrm{T}}\mathrm{y} \tag{3.43}$$

Now even if X is not full rank, the problem is non singular (the inverse exists). In case of orthonormal inputs, the ridge coefficients are the same of least square but scaled: $\hat{\beta}^{\mathrm{ridge}} = \frac{\hat{\beta}}{1+\lambda}$.

The parameter λ can also be derived assuming a prior distribution $\mathrm{y_i} \sim \mathcal{N}\left(\beta_0 + \mathrm{x_i^T}\beta, \sigma^2\right)$ and the parameters β_{j} are distributed as $\mathcal{N}\left(0, \tau^2\right)$. Then from 3.40 $\lambda = \frac{\sigma^2}{\tau^2}$.

Applying the SVD decomposition of the matrix $\mathbf{X} = \mathbf{UDV^T}$ where X is N × p, U is N × pand V is p × p, the latter two both orthogonal with the columns of U spanning the column space of X and the columns of V spanning the row space of X. D is a p × p diagonal matrix with the elements $\mathrm{d_1} \geq \mathrm{d_2} \geq \cdots \geq \mathrm{d_p} \geq 0$ called singular value decomposition of X. If any $\mathrm{d_j} = 0$ then X is singular.

The least squares equation can be rewritten as

$$\mathbf{X}\hat{\beta}^{\mathrm{ls}} = \mathbf{X}\left(\mathbf{X^T X}\right)^{-1}\mathbf{X^T y} = \mathbf{UU^T}\mathrm{y} \tag{3.44}$$

In case of the ridge regression, the coefficients are

$$\begin{aligned}\mathbf{X}\hat{\beta}^{\mathrm{ridge}} &= \mathbf{X}\left(\mathbf{X^T X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X^T y} = \mathbf{UU^T y} = \\ &= \sum_{\mathrm{j}-1}^{\mathrm{p}} \mathbf{u_j} \frac{\mathrm{d_j^2}}{\mathrm{d_j^2} + \lambda} \mathbf{u_j}^{\mathrm{T}}\mathrm{y}\end{aligned} \tag{3.45}$$

where $\mathrm{u_j}$ are the column vectors. So ridge regression first computes the coordinates of y with respect to the orthonormal basis **U**, it then shrinks those coordinates since $\lambda \geq 0$. A greater amount of shrinkage is applied to the coordinates of basis vector with smaller $\mathrm{d_j}$, corresponding to elements with small variance.

3.10.2 Lasso regression

The lasso regression is similar to ridge regression but it uses a L_1 penalization instead of L_2.

$$\hat{\beta}^{\text{lasso}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\} \tag{3.46}$$

or equivalently

$$\begin{aligned} \hat{\beta}^{\text{lasso}} &= \arg\min_{\beta} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \right\} \\ &\text{subject to} \sum_{j=1}^{p} |\beta_j| \leq t \end{aligned} \tag{3.47}$$

If $t > t_0 = \sum_1^p |\beta_j|$ where β_j are the least square coefficients, then the lasso coefficients are the same of the least squares ones. If $t = t_0/2$ then the least square coefficients are shrunk by 50% on average. Making t sufficiently big will cause some of the coefficients to be exactly 0.

In 3.1, the blue areas show the constraints for the estimates of ridge and lasso regressions, while the ellipses are contours of the residual sum of squares, centered at β^{ls}, in case of only two coefficients. For the lasso one of the coefficients is 0 when it hits one of the corner. In higher dimensions the figure becomes a rhomboid with many corners (and faces) so it becomes easy to hit a corner.

Also other penalization factors can be chosen, 3.2

3.10.3 Least angle regression

It is similar to forward stepwise. At first it identifies the variable most correlated with the response but instead of fitting it completely, it moves the variable toward its least squares value. As soon as another value gets correlated to the residual as the previous variable, the process is paused. The second variable joins the active set and their coefficients are moved together in a way that keeps their correlations tied and decreasing. The process is continued until all the variables are in the model. After $\min(N-1, p)$ steps

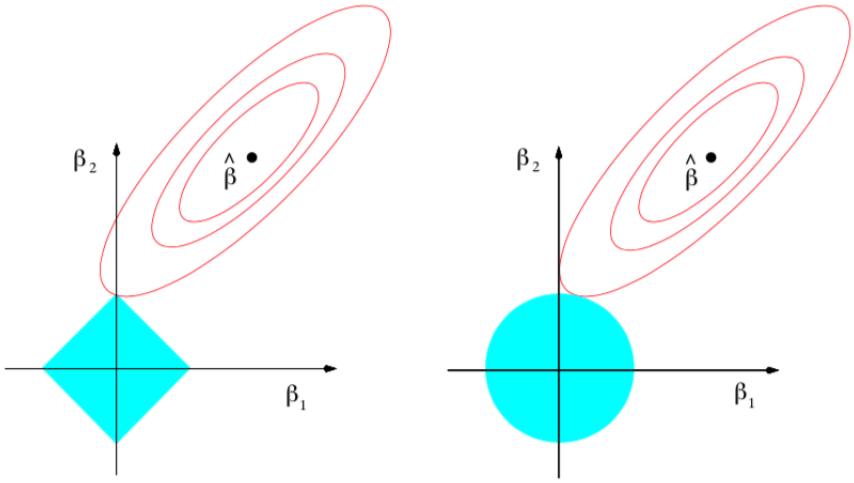


Figure 3.1: Constraints of ridge and lasso regression.

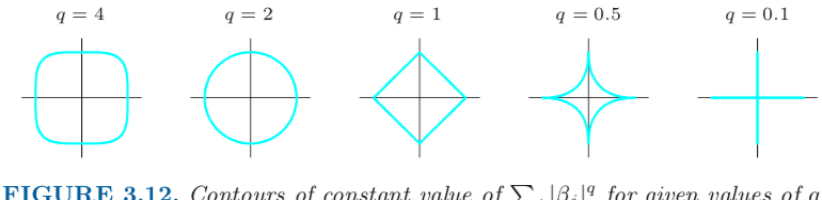


FIGURE 3.12. *Contours of constant value of* $\sum_j |\beta_j|^q$ *for given values of* q.

Figure 3.2: Shapes of different penalization factors.

standardize the predictors to have 0 mean and unit norm;
$\mathcal{A}_k \leftarrow 0$;
$r \leftarrow y - \bar{y}$, $\beta_i = 0$ for $i \neq 0$;
while $|\mathcal{A}_k| < p$ **do**
  find the predictor most correlated to r;
  insert the predictor in the active set $\mathcal{A}_k$;
  move the coefficients of the predictors in the active set to the direction defined by their joint least squares coefficient of the current residual, i.e.,

$$\delta_k = \left(X_{\mathcal{A}_k}^T X_{\mathcal{A}_k}\right)^{-1} X_{\mathcal{A}_k}^T r_k \tag{3.48}$$

  (where $\mathcal{A}_k$ is the current active set of variables) until some other competitor x_l has as much correlation with the current residual;
  $r_k \leftarrow y - X_{\mathcal{A}_k}\beta_{\mathcal{A}_k}$
end

we we arrive at the full least squares solution. The coefficient profile evolves as

$$\beta_{\mathcal{A}_k} = \beta_{\mathcal{A}_k} + \alpha\delta_k \tag{3.49}$$

If to the LAR algorithm we add the following rule i.e.,

If a non-zero coefficient hits 0*, drop its variable from the active set of variables and recompute the current joint least squares direction.*

we get the same coefficient path of the lasso and this is called LAR(lasso). So this become an efficient solution to compute the Lasso problem, especially with $N >> p$ since Lasso can take more than p steps while LAR require p steps and it is efficient since it requires the same complexity as that of a single least squares fit using the p predictors.

3.11 Derived input directions methods

When a large number of inputs is present, often there is a high correlation among them. In this case is convenient to regress on a new set inputs obtained form a linear combination of the original input.

3.11.1 Principal component analysis

First input must be standardized since this analysis depends on the scaling. The principal components are defined as

$$z_i = Xv_i \tag{3.50}$$

where v_i are the column vectors of V from the SVD decomposition of $X = UDV^T$ (recall that z_m are orthogonal). The algorithm then regress X on $z_1, \cdots, z_M$ for $M \leq p$ and we have:

$$\hat{y}^{pcr}_{(M)} = \bar{y}\mathbf{1} + \sum_{m=1}^{M} \hat{\theta}_m z_m \tag{3.51}$$

where $\hat{\theta}_m = \langle z_m, y\rangle / \langle z_m, z_m\rangle$. Since z_m are linear combination of the original predictors x_j, we can express the solution as

$$\hat{\beta}^{pcr}(M) = \sum_{m=1}^{M} \hat{\theta}_m v_m \tag{3.52}$$

With $M = p$ we get the usual least squares. Principal component analysis discards the $p - M$ smallest eigenvalue components.

The value M is suggested by cross-validation.

3.11.2 Partial least squares

This technique use a set of linear combinations of y in addition to X for the construction. It is not scale invariant so each x_j must be standardized.

Since PLS use y to construct its directions, its solution path is not linear in y. It seeks direction with high variance and high correlation with the response while PCR only with high variance.

3.12 Multioutput shrinkage and selection

To apply selection and shrinkage methods in the multiple output case, one could apply a univariate technique individually to each outcome or si- multaneously to all outcomes, i.e., different λ in Ridge or Lasso can be used for each output or the same value can be adopted.

standardize x_j to 0 mean and 1 variance;
$\hat{y}^{(0)} \leftarrow \bar{y}\mathbf{1}$;
$x_j^{(0)} \leftarrow x_j$;
for $m = 1, \cdots, p$ **do**
  $z_m = \sum_{j=1}^{p} \hat{\phi}_{mj} x_j^{(m-1)}$ where $\hat{\phi}_{mj} = \langle x_j^{(m-1)}, y \rangle$;
  $\hat{\theta}_m = \frac{\langle z_m, y \rangle}{\langle z_m, z_m \rangle}$;
  $\hat{y}^m = \hat{y}^{(m-1)} + \hat{\theta}_m z_m$;
  orthogonalize each x_j^{m-1} w.r.t.
    $z_m : x_j^{(m)} = x_j^{(m-1)} - \frac{\langle z_m, x_j^{(m-1)} \rangle}{\langle z_m, z_m \rangle} z_m$, $j = 1, \cdots, p$;
end
Output the sequence of fitted vectors $\{\hat{y}^m\}_1^p$. Since z_{l1}^m are linear in x_j, so is $\hat{y}^{(m)} = X\hat{\beta}^{pls}(m)$. These coefficients can be recovered from the sequence of PLS transformations.

3.13 Other derived algorithms

3.13.1 Incremental forward stagewise

$r \leftarrow y$;
$\beta_i = 0$ for $i \neq 0$;
find the (standardized) predictor x_j most correlated with the residual ;
$\beta_j \leftarrow \beta_j + \delta_j$ where $\delta_j = \epsilon \text{sign}\left[\langle x_j, r \rangle\right]$ and $\epsilon > 0$ small;
$r \leftarrow r - \delta_j x_j$;
Repeat the steps many times until the residuals are uncorrelated with the predictors.

3.13.2 The Dantzig selector

...

3.13.3 The Grouped Lasso

...

4 Bayesian inference

4.1 Introduction to Bayes' theorem

Bayes' theorem is a formula that describes how to update the probabilities of hypotheses when given evidence. It follows simply from the axioms of conditional probability, but can be used to powerfully reason about a wide range of problems involving belief updates.

Given a hypothesis H and evidence E, Bayes' theorem states that the relationship between the probability of the hypothesis Pr(H), before getting the evidence, and the probability of the hypothesis after getting the evidence Pr(H|E) is

$$\Pr(\mathrm{H}|\mathrm{E}) = \frac{\Pr(\mathrm{E}|\mathrm{H})\Pr(\mathrm{H})}{\Pr(\mathrm{E})} \tag{4.1}$$

Often there are competing hypothesis and the task is to determine which is the most probable.

This formula relates the probability of the hypothesis before getting the evidence Pr(H), to the probability of the hypothesis after getting the evidence, Pr(H|E): this term is generally what we want to know. For this reason, Pr(H) is called the **prior probability**, while Pr(H|E) is called the **posterior probability**. The factor that relates the two, $\frac{\Pr(\mathrm{E}|\mathrm{H})}{\Pr(\mathrm{E})}$, is called the **likelihood ratio** and Pr(E|H) is called **likelihood** which indicates the compatibility of the evidence with the given hypothesis. P(H|E) and Pr(E|H) are called conditional probabilities. A conditional probability is an expression of how probable one event is given that some other event occurred (a fixed value) and Bayes' theorem centers on relating different conditional probabilities.

Pr(E) is sometimes called **marginal likelihood** or model evidence this factor is the same for all the hypotheses being considered.

4.2 Bayes inference

Bayesian inference is a method of statistical inference in which Bayes' theorem is used to update the probability for a hypothesis as more evidence or information becomes available. Bayesian inference assumes the data were generated by a model with unknown parameters. From this, it tries to come up with beliefs about the likely "true values" of the parameters of the model. The Bayesian approach differs from the standard ("frequentist") method for inference in its use of a prior distribution to express the uncertainty present before seeing the data, and to allow the uncertainty remaining after seeing the data to be expressed in the form of a posterior distribution.

For this reason when writing down the Bayes theorem in these cases, formally also the conditional on the choice of the model should be written down:

$$\Pr(\theta|\mathbf{X}, \mathrm{M}) = \frac{\Pr(\mathbf{X}|\theta, \mathrm{M})\Pr(\theta|\mathrm{M})}{\Pr(\mathbf{X}|\mathrm{M})} \tag{4.2}$$

Here the hypothesis described in the previous paragraph is represented by a set of values for the parameters.

Once the model is stipulated, $\Pr(\mathrm{X}|\theta)$ can be evaluated for any given set of parameters: in this sense the likelihood is fixed once the model is fixed. $\Pr(\mathbf{X}|\mathrm{M})$ can be rexpressed as $\Pr(\mathbf{X}|\mathrm{M}) = \int \Pr(\mathbf{X}|\theta, \mathrm{M})\Pr(\theta|\mathrm{M})\mathrm{d}\theta$. In this way it can be re-thought as a normalization factor of the sets of parameters since it is a summation over the all parameter space. However note that it **does depend** on the choice of the model. It can be seen also as asking the question *how much is it probable to see the data we have seen given that the model* M*, without any claim about its parameters, generated those data*?

4.3 Types of estimation

It is the moment to clarify different types of estimation. We have already seen the Ordinary Least Square (OLS) estimation, and other types of estimation based on the definition of an error function.

We have also seen the **Maximum Likelihood Estimation (MLE)** approach, which maximizes the likelihood of the Bayes expression and how it is equivalent to OLS estimation in case of linear regression. (To be precise almost always the log-likelihood is maximized, first of all because of analytical convenience since many times we deal with exponentials coming from

Gaussian distribution. Secondly for numerical precision: we are dealing with probabilities, numbers between 0 and 1 and since the range is quite small, underflow might be a problem.) The logarithmic instead extends the range by mapping numbers close to 0 to ∞, resulting in a better precision.

The **Maximum A-Priori (MAP)** estimation maximizes the numerator of the Bayes expression, i.e., the likelihood times the prior, which means the likelihood is weighted by weights coming from the prior. When using a uniform distribution for the prior, MAP turns into MLE since we are assigning equal weights for each possible value. For example suppose we can assign six possible values to β and assume $\mathrm{P}(\beta_\mathrm{i}) = 1/6$:

$$\begin{aligned} \beta_{\mathrm{MAP}} &= \arg\max_{\beta} \sum_{\mathrm{i}} \Pr(\mathbf{x}_\mathrm{i}|\beta) + \log \mathrm{P}(\beta) = \\ &= \arg\max_{\beta} \sum_{\mathrm{i}} \Pr(\mathbf{x}_\mathrm{i}|\beta) + \mathrm{const} = \arg\max_{\beta} \sum_{\mathrm{i}} \Pr(\mathbf{x}_\mathrm{i}|\beta) = \beta_{\mathrm{MLE}} \end{aligned} \tag{4.3}$$

When using a different prior, i.e., the simplification does not hold anymore.

4.4 Conjugate distributions

Definition 4.1. Conjugate distributions In Bayesian probability theory, if the posterior distribution $\Pr(\theta|\mathbf{X}, \mathrm{M})$ is in the same probability space as the prior probability distribution $\Pr(\theta|\mathrm{M})$, then the prior and posterior are called **conjugate distributions**, and the prior is called **conjugate prior for the likelihood function**.

As example consider the Gaussian distribution: **the Gaussian family is conjugate to itself (or self-conjugate) with respect to a Gaussian likelihood function**. If the likelihood is Gaussian, choosing Gaussian prior will ensure that also the posterior distribution is a Gaussian (see 1.6). This means that the Gaussian distribution is a conjugate prior for the likelihood that is also Gaussian.

Consider the general problem of inferring a (continuous) distribution for a parameter θ given some datum or data x. From Bayes' theorem, the posterior distribution is equal to the product of the likelihood function $\mathrm{p}(\mathrm{x}|\theta, \mathrm{M})$ and the prior $\mathrm{p}(\theta|\mathrm{M})$. Let the likelihood function be considered fixed; the likelihood function is usually well-determined from a statement of the data-generating process. It is clear that different choices of the prior distribution

$p(\theta|M)$ may make the integral more or less difficult to calculate, and the product $p(x|\theta, M) \times p(\theta|M)$ may take one algebraic form or another. For certain choices of the prior, the posterior has the same algebraic form as the prior (generally with different parameter values). Such a choice is a conjugate prior. A conjugate prior is an algebraic convenience, giving a closed-form expression for the posterior; otherwise numerical integration may be necessary. Further, conjugate priors may give intuition, by more transparently showing how a likelihood function updates a prior distribution. All members of the exponential family have conjugate priors.

In case of **classification** we are given a training set with input and output data. Once we have stipulated a model that could have generated output data, we want to find the best parameters of the model such that when those inputs are fed to the model we get those outputs. So the question becomes *what is the best set of parameters* θ *for the model* M *that could have generated the output* $\mathbf{y}$ *when the model has been fed with input data* $\mathbf{X}$*?*

In formula:

$$\Pr(\theta|\mathbf{y}, \mathbf{X}, M) = \frac{\Pr(\mathbf{y}|\theta, \mathbf{X}, M)\Pr(\theta|\mathbf{X}, M)}{\Pr(\mathbf{y}|\mathbf{X}, M)} \tag{4.4}$$

4.4.1 Dataset likelihood

4.4.1 To be precise the likelihood is the likelihood of an entire dataset, since we are interested in all $\mathbf{y}$ and not in a single value y. $\Pr(\mathbf{y}|\theta, \mathbf{X}, M)$ is then a joint density over all the responses in our dataset: $\Pr(y_1, y_2, \cdots, y_n|\theta, \mathbf{X}, M)$. Evaluating this density at the observed points gives a single likelihood value for the whole dataset. Assuming that the noise at each data point is independent we can factorize as:

$$\Pr(\mathbf{y}|\theta, \mathbf{X}, M) = \prod_{n=1}^{N} \Pr(y_n|\mathbf{x}_n, \theta) \tag{4.5}$$

We have not say that y_n's are completely independent as otherwise it would not be worth trying to model the data at all. Rather, they are **conditionally independent** given a value θ, i.e., the deterministic model. Basically the model incorporates the dependency. Consider the following example, we have a set of data $(\mathbf{y}, \mathbf{X})$ from which we want to predict the output y_n given a new $\mathbf{x}_n$. Recalling from conditional probability that $P(A|B) = \frac{\Pr(A \cap B)}{\Pr(B)}$ we

have:

$$\mathrm{P}(\mathrm{y_n}|\mathbf{y}, \mathbf{x_n}, \mathbf{X}) = \frac{\Pr(\mathrm{y_n} \cap \mathbf{y}|\mathbf{x_n}, \mathbf{X})}{\Pr(\mathbf{y}|\mathbf{x_n}, \mathbf{X})} \tag{4.6}$$

Note that actually $\mathbf{y}$ does not depend on $\mathbf{x_n}$: $\Pr(\mathbf{y}|\mathbf{x_n}, \mathbf{X}) = \Pr(\mathbf{y}|\mathbf{X})$. Using independence:

$$\mathrm{P}(\mathrm{y_n}|\mathbf{y}, \mathbf{x_n}, \mathbf{X}) = \frac{\Pr(\mathrm{y_n} \cap \mathbf{y}|\mathbf{x_n}, \mathbf{X})}{\Pr(\mathbf{y}|\mathbf{X})} = \frac{\Pr(\mathbf{y}|\mathbf{X})\Pr(\mathrm{y_n}|\mathbf{x_n}, \mathbf{X})}{\Pr(\mathbf{y}|\mathbf{X})} = \tag{4.7}$$

[TO BE CONTINUED WAITING ON ANSWER ON CROSS-VALIDATED STACK EXCHANGE]

5 Linear Classification

For classification problem, in this section we assume the classification boundaries are linear, i.e., in the input hyperspace the points belonging to different classes can be separated by hyperplanes.

5.1 Linear regression of an Indicator matrix

Suppose we have K classes. For a single output we build a vector $\mathbf{y} = (\mathrm{y_1}, \cdots, \mathrm{y_k})$ where $\mathrm{y_k} = 1$ if the class it belongs is the k class. As output we will have the matrix $\mathbf{Y}$ of 0 and 1 with each row having a single 1. We fit a linear regression model to each of the columns of $\mathbf{Y}$ simultaneously:

$$\hat{\mathbf{Y}} = \mathbf{X}\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{Y}$$

We get a coefficient vector for each response column $\mathrm{y_k}$, and hence a $(\mathrm{p}+1)\times\mathrm{K}$ matrix $\hat{\mathbf{B}} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{Y}$,where $\mathbf{X}$ will have $\mathrm{p}+1$ columns with a leading column of 1 for the intercept.

Suppose we are given a new input x. Then the classification problem becomes:

compute the output $\hat{\mathrm{f}}(\mathrm{x})^{\mathrm{T}} = (1, \mathrm{x}^{\mathrm{T}})\hat{\mathbf{B}}$ *which is a* k *vector identify the largest component* $\hat{\mathrm{G}}(\mathrm{x}) = \arg\max_{\mathrm{k}\in\mathcal{G}} \hat{\mathrm{f}}_{\mathrm{k}}(\mathrm{x})$.

With this approach we are basically estimating a conditional expectation, i.e., given the inputs $\mathbf{x}$ what is the probability the output is of class k? Mathematically $\mathbf{E}\left(\mathrm{y_k}|\mathrm{X} = \mathrm{x}\right) = \Pr\left(\mathrm{G} = \mathrm{k}|\mathrm{X} = \mathrm{x}\right)$ since $\mathrm{y_k} = 1$.

Although the linear model guarantees $\sum_{k \in \mathcal{G}} \hat{f}_k = 1$, as long as there is an intercept in the model, $\hat{f}_k(x)$ can be negative or bigger than one, especially when making predictions outside the hull of training data. Although this fact, this approach still works in many cases.

An important limitation is when $K \geq 3$. Even if the classes can still be separated by more than one linear boundaries, linear regression cannot find linear boundaries (because they are more than one?).

Quadratic regression might solve the problem, but a general rule is that if $k \geq 3$ classes are lined up (their centroids are in the same line), a polynomial term with degree up to $k - 1$ is needed to solve the problem, and since the direction is arbitrary, cross-products terms might be needed too.

5.2 Linear Discriminant analysis

For optimal classification we have to know the class posteriors $\Pr(G|X)$. Let π_k be the prior probability of class k with $\sum_k^K \pi_k = 1$. Suppose $f_k(x)$ is the class conditional density. From the Bayes theorem (?) we get

$$\Pr(G = k|X = x) = \frac{\Pr(X = x|G = k)\Pr(G = k)}{\Pr(X = x)} \tag{5.1}$$

$\Pr(G = k)$ is the prior probability π_k, $\Pr(X = x|G = k)$ is the class conditional density while the denominator can be rewritten using the **Total Probability Theorem** as

$$\Pr(X = x) = \sum_{l=1}^{K} \Pr(X = x|G = l)\Pr(G = l) = \sum_{l=1}^{K} \Pr(X = x|G = l)\pi_l \tag{5.2}$$

$$\Rightarrow \Pr(G = k|X = x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l} \tag{5.3}$$

The goodness of classification mostly rely on $f_k(x)$ and many techniques use models for class densities:

- linear and quadratic discriminant analysis use Gaussian densities;
- mixtures of Gaussians allow for non-linear boundaries;
- Naive Bayes models assume that each class density is a product of marginal densities i.e., inputs are conditionally independent in each class.

Modelling each class density as a multivariate Gaussian we have

$$f_k(x) = \frac{1}{(2\pi)^{\frac{p}{2}} |\sum_k|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)} \tag{5.4}$$

Linear Discriminant analysis assumes equal covariance matrices for all classes. Taking as comparison between two classes the log-ratio, we have

$$\begin{aligned}
\log \frac{Pr(G=k|X=x)}{Pr(G=l|X=x)} &= \log \frac{f_k(x)\pi_k}{f_l(x)\pi_l} = \log \frac{\pi_k}{\pi_l} + \log \frac{f_k(x)}{f_l(x)} = \\
&= \log \frac{\pi_k}{\pi_l} + \log \frac{e^{-\frac{1}{2}(x-\mu_k)^T \Sigma^{-1}(x-\mu_k)}}{e^{-\frac{1}{2}(x-\mu_l)^T \Sigma^{-1}(x-\mu_l)}} = \\
&= \log \frac{\pi_k}{\pi_l} - \frac{1}{2}(x-\mu_k)^T \Sigma^{-1}(x-\mu_k) - \left(-\frac{1}{2}\right)(x-\mu_l)^T \Sigma^{-1}(x-\mu_l) = \\
&= \log \frac{\pi_k}{\pi_l} - \frac{1}{2}\left[(x-\mu_k)^T \Sigma^{-1}(x-\mu_k) + (x-\mu_l)^T \Sigma^{-1}(\mu_l - x)\right] = \\
&= \log \frac{\pi_k}{\pi_l} - \frac{1}{2}\left[(2x-\mu_k-\mu_l)^T \Sigma^{-1}(\mu_l - \mu_k)\right] = \\
&= \log \frac{\pi_k}{\pi_l} + \frac{1}{2} x^T \Sigma^{-1}(\mu_k - \mu_l) - \frac{1}{2}(\mu_k + \mu_l)^T \Sigma^{-1}(\mu_k - \mu_l) =
\end{aligned} \tag{5.5}$$

which is linear in x, so all decision boundaries are linear (i.e., they are hyperplanes in p dimensions). If the common covariance matrix is spherical, i.e., $\Sigma = \sigma^2 I$ and the class priors are equal, each boundary that separates two classes is the perpendicular bisector of the segment joining the centroids of the two classes.

The linear discriminant functions of each class are

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2}\mu_k^T + \log \pi_k \tag{5.6}$$

We do not know the parameters of the Gaussian distribution and we must

estimate them from the training data:

$$\hat{\pi}_k = \frac{N_k}{N} \tag{5.7}$$

$$\hat{\mu}_k = \sum_{g_i=k} \frac{x_i}{N_k} \tag{5.8}$$

$$\hat{\Sigma} = \sum_{k=1}^{K} \frac{(x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T}{(N - K)} \tag{5.9}$$

Note that LDA does not use Gaussian assumption for the features.

5.2.1 Decision rule

Consider two classes 1 and 2. LDA classifies to class 1 if

$$x^T \hat{\mathbf{\Sigma}}^{-1} \hat{\mu}_1 - \frac{1}{2} \hat{\mu}_1^T \hat{\mathbf{\Sigma}}^{-1} \hat{\mu}_1 + \log \hat{\pi}_1 > x^T \hat{\mathbf{\Sigma}}^{-1} \hat{\mu}_2 - \frac{1}{2} \hat{\mu}_2^T \hat{\mathbf{\Sigma}}^{-1} \hat{\mu}_2 + \log \hat{\pi}_2 \tag{5.10}$$

to class 2 if $<$ holds. In such case there is a correspondence between LDA and linear regression classification if the two classes are coded with +1 and –1. In this case the coefficient vector from least squares is proportional to the LDA direction. However unless $N_1 = N_2$, the intercepts are different and so are the decision rules.

With more than 2 classes, linear regression is not able to classify correctly while LDA does.

5.3 Quadratic Discriminant analysis

If we do not assume equal covariance, the squared term in x does not cancel out and we get quadratic discriminant functions:

$$\delta_k(x) = -\frac{1}{2} \log |\Sigma_k| - \frac{1}{2} (x - \mu_k)^T \mathbf{\Sigma}_k^{-1} (x - \mu_k) + \log \pi_k \tag{5.11}$$

The decision boundaries between two classes are quadratic functions.

Note: QDA does not differ much from LDA applied the enlarged quadratic polynomial input space but generally QDA is preferred in this case.

The estimates are similar but the covariance matrix must be estimated for each class. When p is large, this means a dramatic increase in the number of parameters, considering we only need the differences $\delta_k(x) - \delta_l(x)$. LDA needs $(K - 1) \times (p + 1)$ parameters, while QDA needs $(K - 1) \times (p\,(p + 3)/2 + 1)$.

5.4 Regularized discriminant analysis

This method shrinks the separate covariances of QDA towards a common covariance as in LDA. In a way it is similar to ridge regression. The regularized covariance matrices have the form:

$$\hat{\mathbf{\Sigma}}_{\mathrm{k}}(\alpha) = \alpha\hat{\mathbf{\Sigma}}_{\mathrm{k}} + (1-\alpha)\hat{\mathbf{\Sigma}} \tag{5.12}$$

with $\alpha \in [0,1]$, the two extremes being LDA and QDA. α can be chosen on the validation data or by cross-validation.

Similarly we can allow $\hat{\mathbf{\Sigma}}$ to be shrunk toward the scalar covariance:

$$\hat{\mathbf{\Sigma}}(\gamma) = \gamma\hat{\mathbf{\Sigma}} + (1-\gamma)\hat{\sigma}^2\mathrm{I} \tag{5.13}$$

with $\gamma \in [0,1]$ so we get a more general family of covariances

$$\hat{\mathbf{\Sigma}}_{\mathrm{k}}(\alpha,\gamma) = \alpha\hat{\mathbf{\Sigma}}_{\mathrm{k}} + (1-\alpha)\left(\gamma\hat{\mathbf{\Sigma}} + (1-\gamma)\,\hat{\sigma}^2\mathrm{I}\right) \tag{5.14}$$

5.5 Computation

Computation of LDA and QDA is simplified by diagonalizing the covariance matrices with the singular value decomposition $\hat{\mathbf{\Sigma}}_{\mathrm{k}} = \mathrm{U_k D_k U_k^T}$. The terms in 5.11 become

$$\begin{aligned}(\mathrm{x}-\mu_{\mathrm{k}})^{\mathrm{T}}\,\mathbf{\Sigma}_{\mathrm{k}}^{-1}\,(\mathrm{x}-\mu_{\mathrm{k}}) &= (\mathrm{x}-\mu_{\mathrm{k}})^{\mathrm{T}}\left(\mathrm{U_k D_k U_k^T}\right)^{-1}(\mathrm{x}-\mu_{\mathrm{k}}) = \\ &= (\mathrm{x}-\mu_{\mathrm{k}})^{\mathrm{T}}\,\mathrm{U_k D_k^{-1} U_k^T}\,(\mathrm{x}-\mu_{\mathrm{k}}) = \left[\mathrm{U_k^T}\,(\mathrm{x}-\mu_{\mathrm{k}})\right]^{\mathrm{T}}\mathrm{D_k^{-1}}\left[\mathrm{U_k^T}\,(\mathrm{x}-\mu_{\mathrm{k}})\right]\end{aligned} \tag{5.15}$$

$$\log|\hat{\mathbf{\Sigma}}_{\mathrm{k}}| = \sum_{\mathrm{l}}\log \mathrm{d_{kl}} \tag{5.16}$$

Considering the above steps, LDA classifier can be seen as performing the following steps:

- sphere the data w.r.t. the common covariance estimate: $\mathrm{X}* \leftarrow \mathrm{D}^{-\frac{1}{2}}\mathrm{U}^{\mathrm{T}}\mathrm{X}$. The common covariance estimate fof X$*$ will now be the identity.
- Classify to the closest centroid in the transformed space, modulo the effect of the class prior probability π_{k}(?)

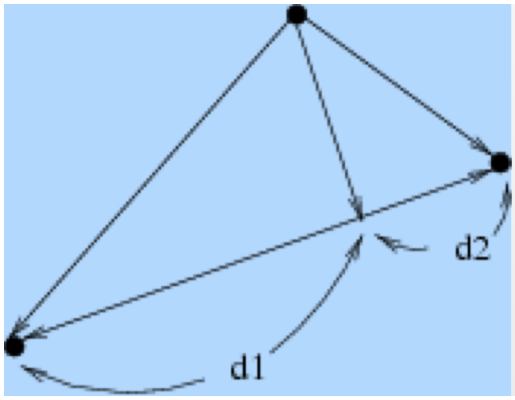


Figure 5.1: ...

5.6 Regularized-rank linear discriminant analysis

Consider K = 2 with two centroids and the input is 2, i.e., input points are on a plane. Given an input point, for classification purposes what matters is not the distance in the p space of such point from the two centroids but rather the distance from the two centroids of the projection of this point on the line joining them (5.1). So basically instead of using the 2 dimensions, we are calculating the distance in one dimension, a line. If K = 3 then the points are projected onto a plane (2d), of course in this case it is convenient if p > 2.

More generally the K centroids in p-dimensional input, lie in an affine subspace of dimensions $\leq$ K – 1 and if p > is much larger than K this will be a considerable drop in dimension.

If K > 3 we can look for a L < K – 1 dimensional subspace optimal for LDA. Fisher defined *optimal* such that the projected centroids were spread out as much as possible in terms of variance. This problem, finding the principal component subspaces of the centroids, involves the following steps:

- compute the k × p matrix of class centroids M and the common covariance matrix W for within-class covariance;
- compute $M* = MW^{-\frac{1}{2}}$ using the eigen-value decomposition;
- compute B∗, the covariance matrix of M∗ (B for between class covariance) and its eigenvalue decomposition $B^* = V^* D_B V^{*T}$. The columns v_l^* from the first to the last define the coordinates of optimal subspaces.

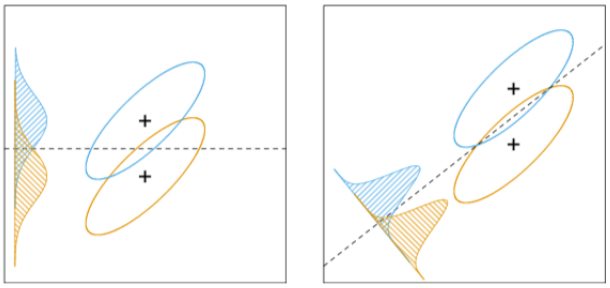


Figure 5.2: Covariance.

The lth discriminant variable is given by $Z_l = v_l^T X$ with $v_l = W^{-\frac{1}{2}} v_l^*$. Although the direction joining the centroids separates the means as much as possible(maximizes the between class covariance), there is an overlap between the projected classes due to the nature of covariances. Taking the covariances into account reduce the overlap and that is what we are doing (5.2).

The between-class variance Z is $a^T Ba$ and the within class variance is $a^T Wa$, with $B + W = T$, the total covariance matrix of X. Fisher's problem maximizes the *Rayleigh quotient*:

$$\max_a \frac{a^T Ba}{a^T Wa} \tag{5.17}$$

This is a generalized eigenvalue problem, with a given by the largest eigenvalue. Similarly one can find the next direction a_2, orthogonal in W to a_1, such that $a_2^T Ba_2 / a_2^T Wa_2$ is maximized; the solution is $a_2 = v_2$, and so on. a_l are the *discriminant coordinates* or *canonical variates*, different from discriminant functions.

The reduced subspaces can be used both for visualization and classification by limiting the distance between centroids to the chosen subspace. However, when doing this, due to the Gaussian classification, a correction factor of $\log \pi_k$ is needed. The misclassification is given by the overlapping area in 5.2 between the two densities. When both classes have the same priors π_k as in the figure, the optimal cut-point is the midway between projected means, if not the cut-point is moved towards the smaller class to have a better error rate.

For 2 classes one can derive the linear rule using LDA, and then choosing the cut-point to minimize misclassification error.

5.7 Logistic regression

The idea behind logistic regression is to still exploit a linear model $x^T\beta$ but having its output representing a probability, i.e., constrained between 0 and 1. This part is performed using the sigmoid function

$$p = \sigma(q) = \frac{1}{1 + e^{-q}} \tag{5.18}$$

Inverting the terms we get

$$q = -\log \frac{1-p}{p} = \log \frac{p}{1-p} \tag{5.19}$$

5.19 is called **logit function**. As q increases to ∞, the output of the sigmoid gets closer to 1; instead when it diverges to $-\infty$ we get 0. Suppose we have just 2 classes or equivalently a binary classifier that tells the probability of an event to happen: $Y_n = 1$ when the event happens and $Y_n = 0$ when it does not. We can express the probabilities output by our classifier when new input data x_{new} are observed as:

$$\begin{aligned} P(G = 1|x_n, \beta) &= \frac{1}{1 + e^{-\beta^T \mathbf{x}_{new}}} \\ P(G = 0|x_n, \beta) &= 1 - \frac{1}{1 + e^{-\beta^T \mathbf{x}_{new}}} = \frac{e^{-\beta^T \mathbf{x}_{new}}}{1 + e^{-\beta^T \mathbf{x}_{new}}} \end{aligned} \tag{5.20}$$

These equations can be combined in a single equation:

$$P(G = g|x_n, \beta) = P(G = 1|x_n, \beta)^g P(G = 0|x_n, \beta)^{g-1} \tag{5.21}$$

Taking the log-ratio between the two probabilities we have:

$$\log \frac{P(Y_n = 0|x_n, \beta)}{P(Y_n = 1|x_n, \beta)} = \log \frac{\frac{e^{-\beta^T \mathbf{x}_{new}}}{\cancel{1+e^{-\beta^T \mathbf{x}_{new}}}}}{\frac{1}{\cancel{1+e^{-\beta^T \mathbf{x}_{new}}}}} = -\beta^T \mathbf{x}_{new} \tag{5.22}$$

So we are using lines (or hyperplanes) to separate the two classes.

5.7.1 Multinomial logistic regression: more than 2 classes

Now suppose we have more than two classes and we still want to separate those classes with linear functions, which means we will have a hyperplane separating two classes. We have K possible outcomes. We can think to run $\mathrm{K}-1$ independent binary logistic regressions with one class chosen as **pivot**, generally the one corresponding to class K, and with the other $\mathrm{K}-1$ classes separately regressed against the pivot:

$$\begin{aligned}
&\log \frac{\Pr(\mathrm{G}=1|\mathrm{X}=\mathrm{x})}{\Pr(\mathrm{G}=\mathrm{K}|\mathrm{X}=\mathrm{x})} = \beta_1^{\mathrm{T}}\mathrm{x} \\
&\log \frac{\Pr(\mathrm{G}=2|\mathrm{X}=\mathrm{x})}{\Pr(\mathrm{G}=\mathrm{K}|\mathrm{X}=\mathrm{x})} = \beta_2^{\mathrm{T}}\mathrm{x} \\
&\vdots \\
&\log \frac{\Pr(\mathrm{G}=\mathrm{K}-1|\mathrm{X}=\mathrm{x})}{\Pr(\mathrm{G}=\mathrm{K}|\mathrm{X}=\mathrm{x})} = \beta_{(\mathrm{K}-1)}^{\mathrm{T}}\mathrm{x}
\end{aligned} \tag{5.23}$$

The choice of the pivot class used as denominator is arbitrarily and the estimates are equivalent under different choices.

Summing the probability of each class we must get 1:

$$\begin{aligned}
&\sum_{\mathrm{l}=1}^{\mathrm{K}} \Pr(\mathrm{G}=\mathrm{l}|\mathrm{X}=\mathrm{x}) = 1 \Rightarrow \Pr(\mathrm{G}=\mathrm{K}|\mathrm{X}=\mathrm{x}) + \sum_{\mathrm{l}=1}^{\mathrm{K}-1} \Pr(\mathrm{G}=\mathrm{l}|\mathrm{X}=\mathrm{x}) = 1 \\
&\Rightarrow \Pr(\mathrm{G}=\mathrm{K}|\mathrm{X}=\mathrm{x}) + \sum_{\mathrm{l}=1}^{\mathrm{K}-1} \Pr(\mathrm{G}=\mathrm{K}|\mathrm{X}=\mathrm{x})\, \mathrm{e}^{\beta_{\mathrm{l}}^{\mathrm{T}}\mathrm{x}} = 1 \\
&\Rightarrow \Pr(\mathrm{G}=\mathrm{K}|\mathrm{X}=\mathrm{x}) = \frac{1}{1+\sum_{\mathrm{l}=1}^{\mathrm{K}-1} \mathrm{e}^{\beta_{\mathrm{l}}^{\mathrm{T}}\mathrm{x}}}
\end{aligned} \tag{5.24}$$

So we can re-express the probabilities as:

$$\Pr(\mathrm{G}=\mathrm{k}|\mathrm{X}=\mathrm{x}) = \frac{\mathrm{e}^{\beta_{\mathrm{k}}^{\mathrm{T}}\mathrm{x}}}{1+\sum_{\mathrm{l}=1}^{\mathrm{K}-1} \mathrm{e}^{\beta_{\mathrm{l}}^{\mathrm{T}}\mathrm{x}}} \tag{5.25}$$

Softmax function We do not have anymore the sigmoid function, instead we have used another function named **softmax** that as the sigmoid takes

any real value as input and outputs a value between 0 and 1. The difference mostly relies on the denominator used for normalization factor since the sigmoid is a 2D curve while the softmax can have higher dimensionality (for 2D it corresponds to the sigmoid).

5.7.2 Fitting logistic regression

From now on we will consider a logistic regression for just two classes.

As seen in 4.4, we want to find the best parameters for the chosen model according to some criteria. First let us apply the Bayes theorem:

$$\Pr\left(\beta|\mathbf{y},\mathbf{X},\mathrm{M}\right)=\frac{\Pr\left(\mathbf{y}|\beta,\mathbf{X},\mathrm{M}\right)\Pr(\beta,\mathrm{M})}{\Pr(\mathbf{y}|\mathbf{X},\mathrm{M})} \tag{5.26}$$

From now on we will not write the conditional on the model for seek of brevity but we know it exists.

This formula tells we want to find the coefficients given some input and output data. Let us analyse each term:

- $\Pr(\beta)$ (**prior distribution**): this is the prior belief about the parameters without seeing the data. As prior, we will use a Gaussian distribution with 0 mean: $\mathcal{N}(0,\sigma^2\mathbf{I})$. In this way the parameters will depend on σ^2: more formally we should write $\Pr(\beta|\sigma)$ but we will skip. Although the choice of the Gaussian distribution with 0 mean is analytically convenient as seen in 4.4, we will not rely on conjugacy.

- $\Pr\left(\mathbf{y}|\beta,\mathbf{X}\right)$ (**likelihood**): we will assume y are conditionally independent (4.4.1):

  $$\Pr(\mathbf{y}|\mathbf{X},\beta)=\prod_{\mathrm{n}=1}^{\mathrm{N}}\Pr(\mathrm{y_n}|\mathbf{x}_\mathrm{n},\beta) \tag{5.27}$$

  Since in this case we have a binary variable, instead of a Gaussian random variable, suitable for real values distribution, we will consider a binary random variable $\mathrm{Y_n}$ characterised by the probability of the second class:

  $$\Pr(\mathbf{y}|\mathbf{X},\beta)=\prod_{\mathrm{n}=1}^{\mathrm{N}}\Pr(\mathrm{Y_n}=\mathrm{y_n}|\mathbf{x}_\mathrm{n},\beta) \tag{5.28}$$

- $\Pr(\mathbf{y}|\mathbf{X})$(**marginal likelihood**): It can be expressed as

$$\Pr(\mathbf{y}|\mathbf{X}) = \int \Pr(\mathbf{y}|\beta, \mathbf{X}) \Pr(\beta) \mathrm{d}\beta \tag{5.29}$$

So the numerator can be calculated and results in a Gaussian function not in standard form (1.6). The denominator, or marginal likelihood, as we have seen can be expressed as $\Pr(\mathbf{y}|\mathbf{X}) = \int \Pr(\mathbf{y}|\beta, \mathbf{X}) \Pr(\beta) \mathrm{d}\beta$. Mathematically this integral has no close solution since (again see 1.6) and the impossibility of the integral of $\mathrm{e}^{-\mathrm{x}^2}$), unless an ad-hoc prior distribution is chosen.

When we cannot directly compute the posterior density (due to the denominator), we have three options:

1. find the single value β that correspond to the highest value of the posterior. This is equivalent to find the value β that maximize the numerator since the denominator is not a function of β but a numerical value.

2. Approximate $\Pr(\beta|\mathbf{X}, \mathbf{y})$ with some other density that we can compute analytically.

3. Sample directly from the posterior $\Pr(\beta|\mathbf{X}, \mathbf{y})$ knowing only the numerator.

The first method is simple and hence popular but it is not very "Bayesian", since we will make predictions of new data based on a single value and not a distribution. The whole Bayesian theory is based on making an hypothesis and getting feedback from the data that can validate or not that hypothesis. Also when using a distribution we get also measures on how confident we are about the estimated model. For example when estimating a Gaussian distribution, we estimate the coefficients μ and σ^2: the smallest the latter value the more confident we are about the model. When using a single value we loose this piece of information.

With the second method we get a density easy to work with but if the chosen density is very different from the posterior our model will not be very reliable.

The third method samples from the posterior and hence to get a good approximation but it can be difficult.

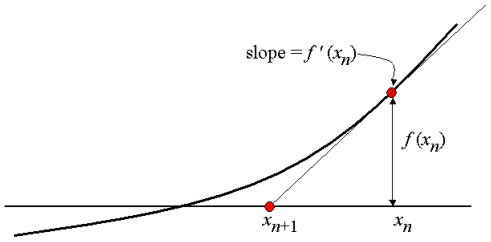


Figure 5.3: Newton-Raphsone example.

5.7.3 First method: point estimation, the MAP solution

We have seen that in some cases we cannot compute the posterior but we can compute the numerator of the expression, i.e., the prior multiplied by the likelihood. The value that maximizes the posterior is also the value that maximizes the numerator. We have already seen a likelihood maximization procedure in **??**; here we are maximizing the likelihood times the prior. This solution is the **maximum a posteriori (MAP)** estimate.

The Newton-Raphson method The Newton-Raphson method finds the points where $f(x) = 0$. Starting from an estimation x_n of such points, the estimation is updated by moving to the point where the tangent to the function at x_n passes throw the x-axis. This point is computed by approximating the gradient as a change in $f(x)$ divided by a change in x:

$$\begin{aligned} f'(x_n) &= -\frac{f(x_n) - 0}{x_n - x_{n+1}} \\ \Rightarrow x_{n+1} &= x_n - \frac{f(x_n)}{f'(x_n)} \end{aligned} \tag{5.30}$$

Instead of finding the point for which $f(x) = 0$, we want to find the point where its derivative is 0. Hence we substitute $f(x)$ with $f'(x)$ and $f'(x)$ with $f''(x)$. In this way the method can be used to find points where the gradient passes through 0, i.e., minima, maxima and points of inflections:

$$\begin{aligned} f''(x_n) &= -\frac{f'(x_n) - 0}{x_n - x_{n+1}} \\ \Rightarrow x_{n+1} &= x_n - \frac{f'(x_n)}{f''(x_n)} \end{aligned} \tag{5.31}$$

When dealing with vectors, $f'(x)$ is replaced by the vector of partial derivatives evaluated at x_n and $\frac{1}{f''(x_n)}$ is replaced by the inverse of the Hessian matrix $\frac{-\partial^2 f(x)}{\partial \mathbf{x} \partial \mathbf{x}^T}$.

Derivation As already done, instead of maximizing the function itself, we maximize its logarithmic. Apart from the mathematical convenience this is also the advantage of avoiding underflow: we are dealing with probabilities, number between 0 and 1 and these might be too small to have a sufficient numerical precision. The logarithmic, as the numbers go to 0, makes the number goes to infinity, guaranteeing a better numerical precision. Since we cannot compute the derivative and set it to 0, we use the Newton-Raphson procedure. Let us call the numerator $g(\beta, \mathbf{X}, \mathbf{y}) = \Pr(\mathbf{y}|\beta, \mathbf{X})\Pr(\beta|\mathbf{X})$:

$$\beta' = \beta - \left(\frac{\partial^2 \log g(\beta, \mathbf{X}, \mathbf{y})}{\partial \beta \partial \beta^T} \right)^{-1} \frac{\partial \log g(\beta, \mathbf{X}, \mathbf{y})}{\partial \beta} \tag{5.32}$$

The procedure is iterative and stops when the gradient is 0. To check the point we have converged to corresponds to a minimum we check the Hessian matrix is negative definite.

Before computing the derivatives we re-express $g(\beta, \mathbf{X}, \mathbf{y})$.

$$\begin{aligned} &\log g(\beta, \mathbf{X}, \mathbf{y}) = \sum_{n=1}^{N} \log \Pr(Y_n = y_n|\mathbf{x}_n, \beta) + \log \Pr(\beta|\sigma^2) = \\ &= \sum_{n=1}^{N} \log \left[\left(\frac{1}{1 + e^{-\beta^T \mathbf{x}_n}} \right)^{y_n} \left(\frac{e^{-\beta^T \mathbf{x}_n}}{1 + e^{-\beta^T \mathbf{x}_n}} \right)^{1-y_n} \right] + \log \Pr(\beta|\sigma^2) = \end{aligned} \tag{5.33}$$

We denote $P_n = P(Y_n = 1|\beta, \mathbf{x}_n) = \frac{1}{1+e^{-\beta^T \mathbf{x}_n}}$:

$$\log g(\beta, \mathbf{X}, \mathbf{y}) = \sum_{n=1}^{N} \log P_n^{y_n} + \log(1 - P_n)^{1-y_n} + \log \Pr(\beta|\sigma^2) = \tag{5.34}$$

Racalling

$$\begin{aligned}\Pr(\beta|\sigma^2) =& \frac{1}{\sqrt{(2\pi)^{\mathrm{D}}|\Sigma|}} \mathrm{e}^{-\frac{(\beta-\mu)^{\mathrm{T}}(\beta-\mu)}{2\sigma 2}} \\ \Rightarrow \log \mathrm{g}(\beta, \mathbf{X}, \mathbf{y}) =& \sum_{\mathrm{n}=1}^{\mathrm{N}} \log \mathrm{P}_{\mathrm{n}}^{\mathrm{y_n}} + \log(1-\mathrm{P_n})^{1-\mathrm{y_n}} - \frac{\mathrm{D}}{2}\log(2\pi) - \mathrm{D}\log\sigma + \\ & - \frac{1}{2\sigma^2}\beta^{\mathrm{T}}\beta\end{aligned} \tag{5.35}$$

Now we can take the derivative:

$$\begin{aligned}\frac{\partial \log \mathrm{g}(\beta, \mathbf{X}, \mathbf{y})}{\partial \beta} =& -\frac{1}{\sigma^2}\beta + \sum_{\mathrm{n}=1}^{\mathrm{N}} \frac{\mathrm{y_n}}{\mathrm{P_n}} \frac{\partial \mathrm{P_n}}{\partial \beta} + \frac{1-\mathrm{y_n}}{1-\mathrm{P_n}} \frac{\partial(1-\mathrm{P_n})}{\partial \beta} = \\ =& \frac{\partial \log \mathrm{g}(\beta, \mathbf{X}, \mathbf{y})}{\partial \beta} = -\frac{1}{\sigma^2}\beta + \sum_{\mathrm{n}=1}^{\mathrm{N}} \frac{\mathrm{y_n}}{\mathrm{P_n}} \frac{\partial \mathrm{P_n}}{\partial \beta} - \frac{1-\mathrm{y_n}}{1-\mathrm{P_n}} \frac{\partial \mathrm{P_n}}{\partial \beta}\end{aligned} \tag{5.36}$$

Now we must calculate $\frac{\partial \mathrm{P_n}}{\partial \beta}$:

$$\begin{aligned}\frac{\partial \mathrm{P_n}}{\partial \beta} =& \frac{\partial}{\partial \beta} \frac{1}{1+\mathrm{e}^{-\beta^{\mathrm{T}}\mathbf{x_n}}} = \frac{\partial}{\partial \beta}\left(1+\mathrm{e}^{-\beta^{\mathrm{T}}\mathbf{x_n}}\right)^{-1} = \\ =& \frac{1}{\left(1+\mathrm{e}^{-\beta^{\mathrm{T}}\mathbf{x_n}}\right)^2} \frac{\partial}{\partial \beta}\left(1+\mathrm{e}^{-\beta^{\mathrm{T}}\mathbf{x_n}}\right) = \\ =& -\mathbf{x_n} \frac{\mathrm{e}^{-\beta^{\mathrm{T}}\mathbf{x_n}}}{\left(1+\mathrm{e}^{-\beta^{\mathrm{T}}\mathbf{x_n}}\right)^2} = -\mathrm{x_n} \frac{1}{\left(1+\mathrm{e}^{-\beta^{\mathrm{T}}\mathbf{x_n}}\right)} \frac{\mathrm{e}^{-\beta^{\mathrm{T}}\mathbf{x_n}}}{\left(1+\mathrm{e}^{-\beta^{\mathrm{T}}\mathbf{x_n}}\right)} = \\ =& -\mathbf{x_n}\mathrm{P_n}(1-\mathrm{P_n})\end{aligned} \tag{5.37}$$

Substituting in 5.36:

$$\begin{aligned}\frac{\partial \log \mathrm{g}(\beta, \mathbf{X}, \mathbf{y})}{\partial \beta} &= -\frac{1}{\sigma^2}\beta + \sum_{\mathrm{n}=1}^{\mathrm{N}} \frac{\mathrm{y_n}}{\mathrm{P_n}} \frac{\partial \mathrm{P_n}}{\partial \beta} - \frac{1-\mathrm{y_n}}{1-\mathrm{P_n}} \frac{\partial \mathrm{P_n}}{\partial \beta} = \\ &= -\frac{1}{\sigma^2}\beta + \sum_{\mathrm{n}=1}^{\mathrm{N}} \frac{\mathrm{y_n}}{\mathrm{P_n}} \left[-\mathbf{x_n}\mathrm{P_n}(1-\mathrm{P_n})\right] + \\ &\quad - \frac{1-\mathrm{y_n}}{1-\mathrm{P_n}} \left[-\mathbf{x_n}\mathrm{P_n}(1-\mathrm{P_n})\right] = \\ &= -\frac{1}{\sigma^2}\beta + \sum_{\mathrm{n}=1}^{\mathrm{N}} -\mathrm{y_n}\mathbf{x_n}(1-\mathrm{P_n}) - (1-\mathrm{y_n})(-\mathbf{x_n}\mathrm{P_n}) = \\ &= -\frac{1}{\sigma^2}\beta + \sum_{\mathrm{n}=1}^{\mathrm{N}} \mathbf{x_n}(\mathrm{y_n}-\mathrm{P_n})\end{aligned} \tag{5.38}$$

Now we must compute the Hessian matrix:

$$\begin{aligned}&\frac{\partial^2 \log \mathrm{g}(\beta, \mathbf{X}, \mathbf{y})}{\partial \beta \partial \beta^{\mathrm{T}}} = \frac{\partial}{\partial \beta^{\mathrm{T}}} \left(-\frac{1}{\sigma^2}\beta + \sum_{\mathrm{n}=1}^{\mathrm{N}} \mathbf{x_n}(\mathrm{y_n}-\mathrm{P_n}) \right) = \\ &= -\frac{1}{\sigma^2}\mathbf{I} - \sum_{\mathrm{n}=1}^{\mathrm{N}} \mathbf{x_n} \frac{\partial \mathrm{P_n}}{\partial \beta^{\mathrm{T}}} = -\frac{1}{\sigma^2}\mathbf{I} - \sum_{\mathrm{n}=1}^{\mathrm{N}} \mathbf{x_n}\mathbf{x_n^{\mathrm{T}}}\mathrm{P_n}(1-\mathrm{P_n})\end{aligned} \tag{5.39}$$

where we used the result from 5.37.

Note that the 2 terms are negative definite, hence the Hessian is always negative definite. Therefore there can only be one optimum and it must be the minimum.

The decision boundary is the one for which $\Pr(\mathrm{Y_n} = 1|\mathbf{x}, \hat{\beta} = 0.5)$.

The steps above can be done for any prior and likelihood combination. In some cases the posterior might have several maxima and or some minima and it become difficult to know if the maximum is a global optimum.

The *Elements of Statistical Learning* book instead of maximizing posterior (or equivalently the numerator), it maximizes the likelihood. Let us express $\Pr(\mathrm{G} = \mathrm{k}|\mathrm{X} = \mathrm{x}) = \mathrm{p_k}(\mathrm{x}, \theta)$ with $\theta = \left\{\beta_{10}, \beta_1^{\mathrm{T}}, \cdots, \beta_{(\mathrm{K}-1)0}, \beta_{\mathrm{K}-1}^{\mathrm{T}}\right\}$

and let us define the log-likelihood

$$\ell(\theta) = \sum_{i=1}^{N} \log p_{g_i}(x_i, \theta) \tag{5.40}$$

Consider just two classes with responses 0, 1 and let $p_1(x, \theta) = p(x, \theta)$ and $p_2(x, \theta) = 1 - p(x, \theta)$. Recall that having two classes $\sum_{l=1}^{K-1} e^{\beta_{l0} + \beta_l^T x_i} = e^{\beta_{10} + \beta_1^T x_i}$. The log-likelihood can be written as:

$$\begin{aligned}
\ell(\beta) &= \sum_{i=1}^{N} \{y_i \log p(x_i, \beta) + (1 - y_i) \log(1 - p(x_i, \beta))\} = \\
&= \sum_{i=1}^{N} y_i \log \frac{e^{\beta_{10} + \beta_1^T x_i}}{1 + \sum_{l=1}^{K-1} e^{\beta_{l0} + \beta_l^T x_i}} + \\
&\quad + \sum_{i=1}^{N} (1 - y_i) \log \left(1 - \frac{e^{\beta_{10} + \beta_1^T x_i}}{1 + \sum_{l=1}^{K-1} e^{\beta_{l0} + \beta_l^T x_i}}\right) = \\
&= \sum_{i=1}^{N} y_i \log \frac{e^{\beta_{10} + \beta_1^T x_i}}{1 + e^{\beta_{10} + \beta_1^T x_i}} + \\
&\quad + \sum_{i=1}^{N} (1 - y_i) \log \left(1 - \frac{e^{\beta_{10} + \beta_1^T x_i}}{1 + e^{\beta_{10} + \beta_1^T x_i}}\right) = \\
&= \sum_{i=1}^{N} y_i \left(\log e^{\beta_{10} + \beta_1^T x_i} - \log\left(1 + e^{\beta_{10} + \beta_1^T x_i}\right)\right) + \\
&\quad + \sum_{i=1}^{N} (1 - y_i) \log \frac{1}{1 + e^{\beta_{10} + \beta_1^T x_i}} = \\
&= \sum_{i=1}^{N} y_i \left(\log e^{\beta_{10} + \beta_1^T x_i} - \log\left(1 + e^{\beta_{10} + \beta_1^T x_i}\right)\right) + \\
&\quad - \sum_{i=1}^{N} (1 - y_i) \log\left(1 + e^{\beta_{10} + \beta_1^T x_i}\right) = \\
&= \sum_{i=1}^{N} \left\{y_i \left(\beta_{10} + \beta_1^T x_i\right) - \log\left(1 + e^{\beta_{10} + \beta_1^T x_i}\right)\right\}
\end{aligned} \tag{5.41}$$

Here $\beta = [\beta_{10}, \beta_1]$. To maximize the log-likelihood we set the derivative to 0:

$$\frac{\partial \ell(\beta)}{\partial \beta} = \sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{x_i}(\mathrm{y_i} - \mathrm{p}(\mathrm{x_i}, \beta) = 0 \tag{5.42}$$

Using the *Newton-Raphson* algorithm that requires the second-derivative:

$$\frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^{\mathrm{T}}} = -\sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{x_i} \mathrm{x_i^T} \mathrm{p}(\mathrm{x_i}, \beta)(1 - \mathrm{p}(\mathrm{x_i}, \beta) \tag{5.43}$$

$$\Rightarrow \beta^{\mathrm{new}} = \beta^{\mathrm{old}} - \left(\frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^{\mathrm{T}}} \right)^{-1} \frac{\partial \ell(\beta)}{\partial \beta} \tag{5.44}$$

Using the matrix notation,

$$\frac{\partial \ell(\beta)}{\partial \beta} = \mathrm{X}^{\mathrm{T}}(\mathrm{y} - \mathrm{p}) \tag{5.45}$$

$$\frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^{\mathrm{T}}} = -\mathrm{X}^{\mathrm{T}}\mathrm{W}\mathrm{X} \tag{5.46}$$

So the Newton step becomes

$$\begin{aligned} \beta^{\mathrm{new}} &= \beta^{\mathrm{old}} + (\mathrm{X}^{\mathrm{T}}\mathrm{W}\mathrm{X})^{-1}\mathrm{X}^{\mathrm{T}}(\mathrm{y} - \mathrm{p}) = \\ &= (\mathrm{X}^{\mathrm{T}}\mathrm{W}\mathrm{X})^{-1}\mathrm{X}^{\mathrm{T}}\mathrm{W}\left(\mathrm{X}\beta^{\mathrm{old}} + \mathrm{W}^{-1}(\mathrm{y} - \mathrm{p})\right) = \\ &= \left(\mathrm{X}^{\mathrm{T}}\mathrm{W}\mathrm{X}\right)^{-1}\mathrm{X}^{\mathrm{T}}\mathrm{W}\mathrm{z} \end{aligned} \tag{5.47}$$

with

$$\mathrm{z} = \mathrm{X}\beta^{\mathrm{old}} + \mathrm{W}^{-1}(\mathrm{y} - \mathrm{p}) \tag{5.48}$$

sometimes known as the adjusted response. This algorithm is known as **iteratively reweighted least squares (IRLS)** since each iteration solves the weighted least square problem:

$$\beta^{\mathrm{new}} \leftarrow \arg\min_{\beta} (\mathrm{z} - \mathrm{X}\beta)^{\mathrm{T}} \mathrm{W} (\mathrm{z} - \mathrm{X}\beta) \tag{5.49}$$

$\beta = 0$ seems a good starting value. Convergence is never guaranteed but typically the algorithm does converge, since the log-likelihood is concave but

overshooting can occur. In the rare cases that the log-likelihood decreases, step size halving will guarantee convergence.

For $K > 2$ we still use the iteration procedure but we will have a $K-1$ vector response and a non-diagonal weight matrix per observation. In this case it is better to work with the vector θ directly.

5.7.4 Second method: Laplace approximation

There are many approximation methods but the most common is the Laplace approximation. The idea is to approximate the density of interest with a Gaussian.

Choosing a Gaussian means choosing a proper variance and mean value. The Laplace approximation method fixes one of the two parameters, i.e., the mean to the value maximizing the posterior, β. We approximate $\log g(\beta, \mathbf{X}, \mathbf{y})$ with the Taylor expansion around $\hat{\beta}$:

$$\begin{aligned}\log g(\beta, \mathbf{X}, \mathbf{y}) \approx & \log g(\hat{\beta}, \mathbf{X}, \mathbf{y}) + \cancel{\frac{\partial \log g(\beta, \mathbf{X}, \mathbf{y})}{\partial \beta}\bigg|_{\hat{\beta}} \left(\beta - \hat{\beta}\right)} + \\ & + \frac{\partial^2 \log g(\beta, \mathbf{X}, \mathbf{y})}{\partial \beta^2}\bigg|_{\hat{\beta}} \left(\beta - \hat{\beta}\right)^2 = \\ & = \log g(\hat{\beta}, \mathbf{X}, \mathbf{y}) + \frac{\partial^2 \log g(\beta, \mathbf{X}, \mathbf{y})}{\partial \beta^2}\bigg|_{\hat{\beta}} \frac{\left(\beta - \hat{\beta}\right)^2}{2}\end{aligned} \tag{5.50}$$

where the second term is the gradient evaluated at the maximum point that therefore must be 0.

The Gaussian density and its log are the following:

$$\begin{aligned}& \frac{1}{2\pi} e^{-\frac{(\beta-\hat{\beta})^2}{2\sigma^2}} \\ \Rightarrow & \log \text{const} - \frac{1}{2\sigma^2}(\beta - \mu)^2\end{aligned} \tag{5.51}$$

which is similar to 5.50. By analogy of the two equations:

$$\begin{aligned}\mu &= \hat{\beta} \\ -\frac{1}{\sigma^2} &= -\frac{\partial^2 \log g(\beta, \mathbf{X}, \mathbf{y})}{\partial \beta^2}\bigg|_{\hat{\beta}}\end{aligned} \tag{5.52}$$

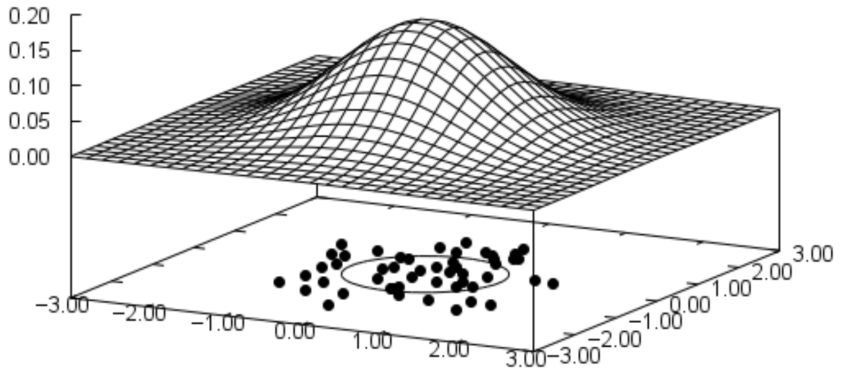


Figure 5.4: Example of multinomial Gaussian.

It can be applied also to multivariate densities $\Pr(\beta, \mathbf{X}, \mathbf{y}) \approx \mathcal{N}(\mu, \mathbf{\Sigma})$:

$$\begin{aligned} \mu &= \hat{\beta} \\ \Sigma^{-1} &= -\frac{\partial^2 \log \mathrm{g}(\beta, \mathbf{X}, \mathbf{y})}{\partial \beta \partial \beta^{\mathrm{T}}}\bigg|_{\hat{\beta}} \end{aligned} \tag{5.53}$$

Consider a multinomial 2D Gaussian function (dimensionaity of β is 2) as in 5.4 and suppose to project on the plane the points of the curve having the same value (the ellipses in the figure). The ellipses will be the combination of coefficients giving the same function value.

In figure 5.5 the Laplace approximation of the posterior (darker lines) is shown while the lighter lines are the true unnormalised posterior. The centre point corresponds to the maximum point $\hat{\beta}$. Note how the approximation is good around $\hat{\beta}$ and it diverges going further from it.

We use the approximate posterior to compute predictions. But now we have a density and not a single value: the prediction is computed by averaging over this density: it is like averaging over all possible values of β. We should calculate the expected value $\Pr(\mathrm{Y}_{\mathrm{new}} = 1|\mathbf{x}_{\mathrm{new}}, \beta)$ with respect to the approximate posterior denoted as $\mathcal{N}(\mu, \mathbf{\Sigma})$:

$$\Pr(\mathrm{Y}_{\mathrm{new}} = 1|\mathbf{x}_{\mathrm{new}}, \mathbf{X}, \beta) = \mathbf{E}_{\mathcal{N}(\mu, \mathbf{\Sigma})} \left\{ \Pr(\mathrm{T}_{\mathrm{new}} = 1|\mathbf{x}_{\mathrm{new}}, \beta) \right\} \tag{5.54}$$

However, we cannot compute the integral (of the expectation), but we can

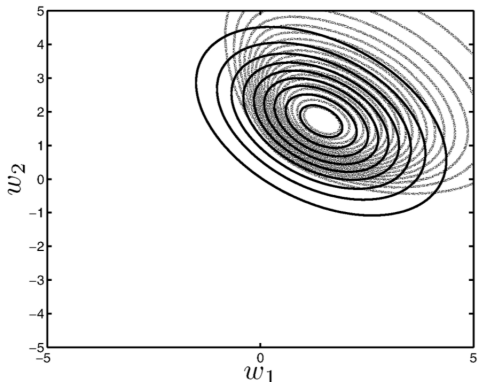


Figure 5.5: The axis are the parameters. The Laplace approximation of the posterior (darker lines) is shown while the lighter lines are the true unnormalised posterior. The centre point corresponds to the maximum point $\hat{\beta}$.

sample from $\mathcal{N}(\mu, \mathbf{\Sigma})$ and approximate the expectation with a sum:

$$\Pr(\mathrm{Y}_{\mathrm{new}} = 1 | \mathbf{x}_{\mathrm{new}}, \mathbf{X}, \beta) = \frac{1}{\mathrm{N_s}} \sum_{\mathrm{s}=1}^{\mathrm{N_s}} \frac{1}{1 + \mathrm{e}^{-\beta_{\mathrm{s}}^{\mathrm{T}} \mathbf{x}_{\mathrm{new}}}} \tag{5.55}$$

Comparison between the decision boundaries of MAP and Laplace method. In case of MAP estimation we get a single separating line as in 5.6a for the example of two classes. In case of Laplace approximation we don't have a single separating line but a density distribution. 5.6b shows 20 boundaries (i.e., set of coefficients) sampled randomly from the distribution. All or almost all separates the classes quite well but in the area from the graph further from the classes (or from the centroids or clusters) there is a lot of variability. This variability represents the uncertainty of the classifier in those area far from the classes, where no event (in the sense of data entry) was observed.

This is made clearer by looking at the decision boundaries in 5.6c and 5.6d. In case of MAP it is quite obvious: the probability increases or decreases just moving close to or far from to the boundary. In case of Laplace approximation the contours are no longer straight lines. The probability are now close to 0.5 in all the area except those closes to the two classes. It is like the classifier is unable to take a decision in those areas, that are the ones

where no event has been noted.

On the contrary the classifier resulting from MAP approximation is always sure about its work: it can only classify either with one or the other class except in on the points on the boundaries. This result from the fact that it is the result of a single point and not a distribution: i.e., the amount of confidence on the result is left out.

5.7.5 Third method: sampling technique

The reason of estimating the posterior density with Laplace approximation or any other method is to take into account uncertainties in β when making predictions. Using the Gaussian approximation we were able to sample as in equation 5.55 directly from data. When deciding we take the conditional probability by averaging all over the potential values β by taking the expectation:

$$\Pr(\mathrm{T_{new}} = 1|\mathrm{x_{new}}, \mathbf{X}, \mathbf{y}, \sigma^2) = \mathbf{E}_{\Pr(\beta|\mathbf{X},\mathbf{y},\sigma^2)}(\Pr(\mathrm{T_{new}} = 1|\mathrm{x_{new}}, \beta) \quad (5.56)$$

In these types of approximations we cut off the approximation step and sample directly from the posterior. A set of samples from the true posterior could be substituted directly into equation 5.55 to compute the desired predictive probability $\Pr(\mathrm{y_i}|\mathbf{x_i}, \mathbf{X}, \beta)$. A popular sampling technique is the **Metropolis-Hastings** algorithm.

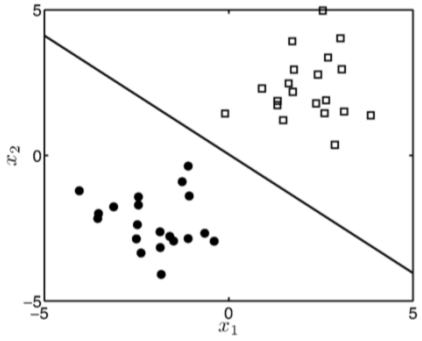
Drawing from the posterior does not mean we can write it down but that we sample directly the system (physical or whatever) generating the data.

The metropolis-Hastings algorithms The objective is to sample from $\Pr(\beta|\mathbf{X}, \mathbf{y}, \sigma^2)$ to approximate the following expectation:

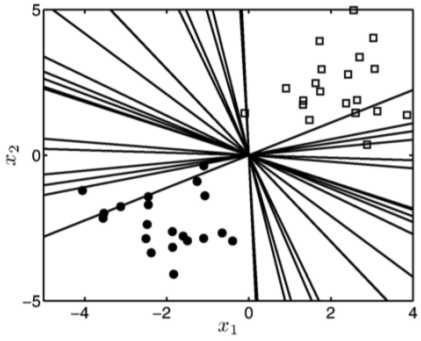
$$\begin{aligned} \Pr(\mathrm{Y_{new}} = 1|\mathrm{x_{new}}, \mathbf{X}, \mathbf{y}, \sigma^2) &= \mathbf{E}_{\Pr(\beta|\mathbf{X},\mathbf{y},\sigma^2)}\left[\Pr(\mathrm{T_{new}} = 1|\mathrm{x_{new}}, \beta)\right] \\ &= \int \Pr(\mathrm{Y_{new}} = 1|\mathrm{x_{new}}, \beta)\Pr(\beta|\mathbf{X}, \mathbf{y}, \sigma^2)\mathrm{d}\beta \end{aligned} \quad (5.57)$$

with

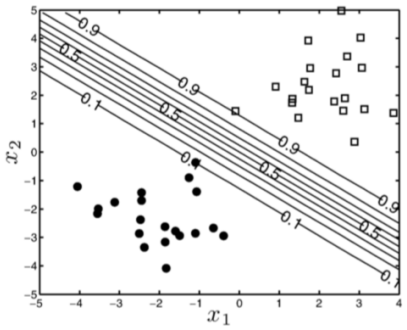
$$\Pr(\mathrm{Y_{new}} = 1|\mathrm{x_{new}}, \mathbf{X}, \mathbf{y}, \sigma^2) \approx \frac{1}{\mathrm{N_s}} \sum_{\mathrm{s}=1}^{\mathrm{N_s}} \Pr(\mathrm{Y_{new}} = 1|\mathrm{x_{new}}, \beta_\mathrm{s}) \quad (5.58)$$



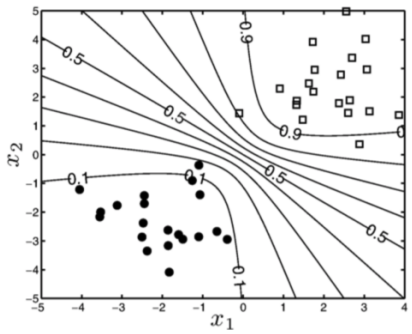
(a) Decision boundary for MAP estimation.



(b) Decision boundary for Laplace approximation estimation.



(c) Contours of probability of belonging to class 1 in case of MAP estimation.



(d) Contours of probability of belonging to class 1 in case of Laplace approximation estimation computed with a sample based approximation.

Figure 5.6: Comparison between MAP and Laplace approximation contours and probability densities.

The algorithm generates a sequence of samples $\beta_1, \beta_2, \cdots$. First all the algorithm is independent from the starting point, as long as we sample long enough: it is guaranteed the sequence converges.

The generation of new samples happens in this way. Suppose we have the sample s – 1, we will propose a new sample $\tilde{\beta}_\mathrm{s}$ and define the density $\Pr(\tilde{\beta}_\mathrm{s}|\beta_{\mathrm{s}-1})$. This density is unrelated with the posterior $\Pr(\beta|\mathbf{X}, \mathbf{y}, \sigma^2)$ and we can define it as we please but it will affect the convergence time. A common choice is to use a Gaussian centred on the current sample, $\beta_{\mathrm{s}-1}$:

$$\Pr(\tilde{\beta}_\mathrm{s}|\beta_{\mathrm{s}-1}, \Sigma) = \mathcal{N}(\beta_{\mathrm{s}-1}, \Sigma) \tag{5.59}$$

Generally σ is taken diagonal with same values. The smaller the elements on the diagonal, the smaller the distance at each step.

Such a sequence creates a **random walk**. The choice of the Gaussian is justified bye the easy of sampling from the Gaussian and by its symmetry: moving from $\tilde{\beta}_{\mathrm{s}-1}$ to $\tilde{\beta}_\mathrm{s}$ is just as likely to move from $\tilde{\beta}_\mathrm{s}$ to $\tilde{\beta}_{\mathrm{s}-1}$:

$$\Pr(\tilde{\beta}_\mathrm{s}|\beta_{\mathrm{s}-1}, \Sigma) = \Pr(\tilde{\beta}_{\mathrm{s}-1}|\beta_\mathrm{s}, \Sigma) \tag{5.60}$$

Accepting or rejecting the candidate $\tilde{\beta}_\mathrm{s}$ is performed by calculating the following quantity:

$$\mathrm{r} = \frac{\Pr(\tilde{\beta}_\mathrm{s}|\mathbf{X}, \mathbf{y}, \sigma^2)}{\Pr(\beta_{\mathrm{s}-1}|\mathbf{X}, \mathbf{y}, \sigma^2)} \frac{\Pr(\beta_{\mathrm{s}-1}|\tilde{\beta}_\mathrm{s}, \Sigma)}{\Pr(\hat{\beta}_\mathrm{s}|\beta_{\mathrm{s}-1}, \Sigma)} \tag{5.61}$$

The expression is the product of the ratio of the posterior density at the proposed sample to that at the old sample times the ratio of the proposed densities. For the Gaussian symmetry discussed above, this term is 1 when using Gaussian densities. Note that we cannot compute exactly the densities, but being a ratio the normalisation constant (the denominator in Bayes expression) simplifies and only the likelihoods times the priors are left:

$$\mathrm{r} = \frac{\Pr(\tilde{\beta}_\mathrm{s}|\sigma^2)}{\Pr(\beta_{\mathrm{s}-1}|\sigma^2)} \frac{\Pr(\mathbf{y}|\mathbf{X}, \tilde{\beta}_\mathrm{s}, \sigma^2)}{\Pr(\mathbf{y}|\mathbf{X}, \beta_{\mathrm{s}-1}, \sigma^2)} \tag{5.62}$$

If $\mathrm{r} > 1$, i.e, we get a higher posterior density, we accept the candidate otherwise we accept the candidate with probability equal to r. The complete algorithm is depicted in 5.7 where a uniform distribution in $[0, 1]$ is used as

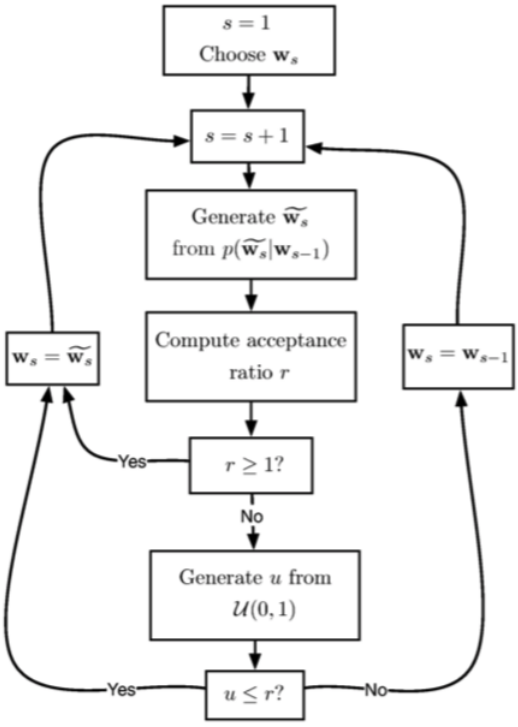
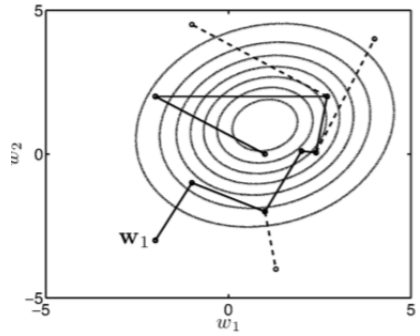


Figure 5.7: The steps of the Metropolis-Hastings algorithm. U is a uniform distribution in $[0, 1]$



(e) After ten samples.

Figure 5.8: Example of Metropolis-Hastings iterations: solid lines are accepted coefficients, dashed lines are the rejected ones.

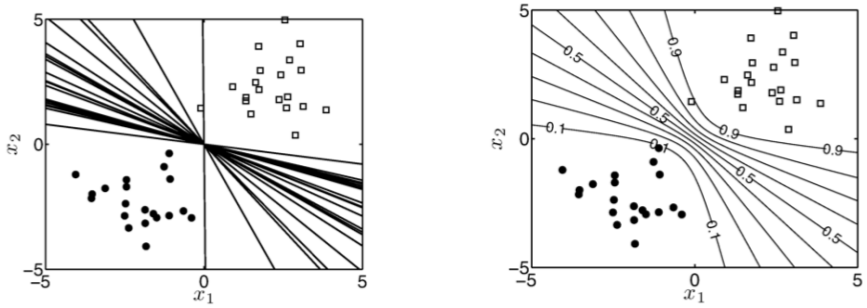
decision rule in case $\mathrm{r} < 1$. Being a uniform distribution, the probability that $\mathrm{u} \leq \mathrm{r}$ is r.

5.8 shows a 10-iteration process of the algorithms where solid lines are accepted coefficients, dashed lines are the rejected ones. Note that even the third sample causes a decrease in the posterior $\mathrm{r} < 1$, nevertheless in this specific case the decision rule accepted it. On the contrary the 4-th sample causes a huge decrease hence it is very unlikely it is accepted and in fact it is not, so $\beta_4 = \beta_3$. After N samples we compute the sample based approximations to the mean and covariance

$$\begin{aligned} \mu' &= \frac{1}{\mathrm{N_s}} \sum_{\mathrm{s}=1}^{\mathrm{N_s}} \beta_{\mathrm{s}} \\ \mathrm{S}' &= \frac{1}{\mathrm{N_s}} \sum_{\mathrm{s}=1}^{\mathrm{N_s}} (\beta_{\mathrm{s}} - \mu')(\beta_{\mathrm{s}} - \mu')^{\mathrm{T}} \end{aligned} \tag{5.63}$$

Definition 5.1. Burn-in It is the time interval between the starting of the algorithm and the convergence.

It cannot be determined: to overcome this problem, a method for determining convergence (to a distribution, not to a value) should be established. For example, in the algorithm above, we do not know if the starting point belongs to an area from which we are supposed to sample: it might be very



(a) Predictive probability contours of classifying objects as square.

(b) Decision boundaries created from randomly selected MH samples.

Figure 5.9: Example of Metropolis-Hastings algorithm results.

far from the posterior. Including these samples in the approximation might result in a not good value. That is why the first samples (ranging from few samples to thousands) should be discarded.

A popular method is to start several samplers simultaneously from different starting points. When all the samplers are generating samples with similar mean and variance, it suggests they converged all to the same distributions.

Definition 5.2. convergence In this case we are talking about the convergence to a given distribution, not a single point. The convergence to a distribution is characterized to the convergence of its parameters: in case of the Gaussian the mean and variance.

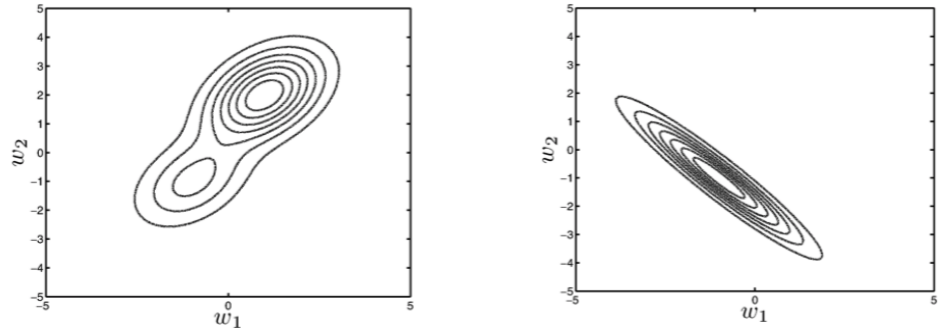
We can even look at each coefficient independently:

$$\Pr(\beta_1|\mathbf{X},\mathbf{y},\sigma^2) = \int \Pr(\beta_1,\beta_2|\mathbf{X},\mathbf{y},\sigma^2)\mathrm{d}\beta_2 \tag{5.64}$$

To calculate the predictive probability using the obtained set of samples, we can do what already done with the Laplace approximation:

$$\Pr(\mathrm{Y}_{\mathrm{new}} = 1|\mathbf{x}_{\mathrm{new}},\mathbf{X},\mathbf{y},\sigma^2) = \frac{1}{\mathrm{N_s}}\sum_{\mathrm{s}=1}^{\mathrm{N_s}} \frac{1}{1+\mathrm{e}^{-\beta_{\mathrm{s}}^{\mathrm{T}}\mathbf{x}_{\mathrm{new}}}} \tag{5.65}$$

5.9a and 5.9b show an example of possible shapes of contours: it does not look too different from the Laplace ones. The only difference is that these



(a) A bi-modal density. (b) Highly correlated parameter density.

Figure 5.10: Example of Metropolis-Hastings algorithm results.

contours are not so tight as the Laplace's ones. This suggests the probability decreases more slowly.

Limitations The difficult mostly lies in the unknown shape of the density. When a density has two or more modes, MH moves towards the modes as these moves increase the posterior density and hence are always accepted. When close to a mode, many steps are required to move from one mode to another and this is very unlikely. We might end up exploring a mode without even knowing the other exists (see 5.10a). Another problem arises when the variables are strongly correlated (see 5.10b). Let us pick any position and propose a movement from a Gaussian with diagonal covariance (i.e., having circular contours). The shapes are very different and many samples will be rejected: the majority of moves that we sample from our proposal will involve moving steeply down the probability gradient. There are even other problems.

5.7.6 Usage

LR is used as data analysis tool where the goal is to understand the role of the input variables in explaining the outcome. Typically many models are fit in a search for a parsimonious model involving a subset of the variables, possibly with some interactions terms.

It is widely used in biostatistical applications where binary responses (two classes) occur quite frequently. For example, patients survive or die, have heart disease or not, or a condition is present or absent.

5.8 Regularized Logistic regression

We can use the L_1 penalty for variable selection and shrinkage:

$$\underset{\beta_0,\beta_1}{\arg\max} \left\{ \sum_{i=1}^{N} y_i \left(\beta_0 + \beta^T x_i\right) - \log\left(1 + e^{\beta_0+\beta^T x_i}\right) - \lambda \sum_{j=-1}^{p} |\beta_j| \right\} \quad (5.66)$$

This function is concave and can be solved using a nonlinear programming method.

5.9 Logistic vs LDA

The difference between the models relies on how the linear coefficients are estimated. The logistic regression model is more general since it makes less assumptions.

LDA is not robust to outliers since observations far from the decision boundary are used to estimate the common covariance matrix, while they are scaled down in the Logistic regression.

5.10 Perceptron learning algorithm

It tries to find a separate hyperplane by minimizing the distance of misclassified points to the decision boundary. If a response $y_i = 1$ is misclassified, then $x_i^T\beta+\beta_0 < 0$, and the opposite for a misclassified response with $y_i = -1$. The goal is to minimize

$$D(\beta, \beta_0) = -\sum_{i\in\mathcal{M}} y_i \left(x_i^T\beta + \beta_0\right) \quad (5.67)$$

where $\mathcal{M}$ is the set of misclassified points. The gradient is

$$\frac{\partial D(\beta, \beta_0}{\partial\beta} = -\sum_{i\in\mathcal{M}} y_i x_i \quad (5.68)$$

$$\frac{\partial D(\beta, \beta_0}{\partial\beta_0} = -\sum_{i\in\mathcal{M}} y_i \quad (5.69)$$

where the algorithm uses the stochastic gradient descent where the coefficients are updated by the gradient value weighted by a step ρ. There are many problems though:

- when data are separable, there are many solutions which depend on the starting value;
- many steps might be required;
- when data are not separable, the algorithm will not converge;

5.11 Optimal separating hyperplanes

The optimal separating hyperplane separates the two classes and maximizes the distance to the closest point from either class (Vapnik, 1996). Not only does this provide a unique solution to the separating hyperplane problem, but by maximizing the margin between the two classes on the training data, this leads to better classification performance on test data.

Suppose M is the distance. Then

$$\begin{aligned} &\max_{\beta,\beta_0,||\beta||=1} \mathrm{M} \\ &\mathrm{y_i}(\mathrm{x_i^T}\beta + \beta_0) \geq \mathrm{M} \end{aligned} \tag{5.70}$$

We can move the constrain on the module $\|||\beta|| = 1$ to the condition $\frac{1}{||\beta||}\mathrm{y_i}(\mathrm{x_i^T}\beta + \beta_0) \geq \mathrm{M}$, which redefines β_0.

For a pair of coefficients satisfying this inequalities, any positively scaled multiple satisfies them too, so we can set arbitrarily set $||\beta|| = \frac{1}{\mathrm{M}}$:

$$\begin{aligned} &\min_{\beta,\beta_0} \frac{1}{2}||\beta||^2 \\ &\textit{subject to } \mathrm{y_i}(\mathrm{x_i^T}\beta + \beta_0) \geq 1 \end{aligned} \tag{5.71}$$

With this constraint, a margin around the decision boundary with thick $\frac{1}{||\beta||}$ is present. We choose β to maximize the thickness of margin margin. This is a convex optimization problem. The Lagrange primal function to be minimized is:

$$\mathrm{L_p} = \frac{1}{2}||\beta||^2 = \sum_{\mathrm{i}=1}^{\mathrm{N}} \alpha_\mathrm{i} \left[\mathrm{y_i}\left(\mathrm{x_i^T}\beta + \beta_0\right)\right] \tag{5.72}$$

and setting the derivatives to 0:

$$\beta = \sum_{\mathrm{i}=1}^{\mathrm{N}} \alpha_\mathrm{i}\mathrm{y_i}\mathrm{x_i}0 \quad = \sum_{\mathrm{i}=1}^{\mathrm{N}} \alpha_\mathrm{i}\mathrm{y_i} \tag{5.73}$$

and substituting in 5.72:

$$\begin{aligned} L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k x_i^T x_k \\ \textit{subject to } \alpha_i \geq 0 \textit{ and } \sum_{i=1}^{N} \alpha_i y_i = 0 \end{aligned} \tag{5.74}$$

The optimal hyperplane focuses more on points that counts, i.e., close to the border. For this reason it is more robust. LDA depends on all the data, even points faraway. If classes are really Gaussian LDA is optimal and separating hyperplane will pay a price for focusing on noisier data. For logistic regression, if a separate hyperplane exists, since the log-likelihood can be driven to 0.

When data are not separable there will be no feasible solution and an alternative formulation is needed.

6 Basis expansion and regularization

To extend the flexibility of linear model, a transformation of $\mathbf{X}$ can be performed and then use a linear model to the derived input features. Denoting the transformation $h_m(\mathbf{X}) : \mathcal{R}^p \rightarrow \mathcal{R}$ for a single input m, we have the model:

$$f(\mathbf{X}) = \sum_{m=1}^{M} \beta_m h_m(\mathbf{X}) \tag{6.1}$$

h() are called **basis function** Any kind of transformation can be applied: square-roots, squared (on single inputs i.e, x_i^2 or among different inputs $x_i x_j$), logarithmic, power functions, trigonometric, etc.

Polynomials are more widely used but they have a limitation due to its global nature: when tweaking the parameters to achieve a desired behaviour in a specific region, problems might arise in another region.

A solution is to consider **piecewise-polynomials** and **splines**.

6.1 Piecewise Polynomials and splines

For the moment X is assumed one-dimensional. A piecewise polynomial function is obtained by dividing the domain of X into contiguous intervals

and use a different f in each interval.

Given a sequence $a = t_0 < t_1 < \cdots < t_{m+1} = b$, the piecewise polynomials are delimited by an adjacent pair in the sequence. So we have the following sequence of intervals:

$$I_l = [t_{l-1}, t_l) \ \textit{for}\ 1 \leq l \leq m \ \textit{and}\ [t_m, t_{m+1}] \tag{6.2}$$

So the piecewise polynomials, having basis functions: $h_0(X) = 1, h_1(X) = X, h_2(X) = X^2$ and so on, is:

$$f(x) = \begin{cases} g_0(x) = \beta_{0,0} + \beta_{0,1}X + \cdots + \beta_{0,k-1}X^{k-1} & X \in I_1 \\ \vdots \\ g_{m-1}(x) = \beta_{m-1,0} + \beta_{m-1,1}X + \cdots + \beta_{m-1,k-1}X^{k-1} & X \in I_{k+1} \end{cases} \tag{6.3}$$

In this way we get discontinuities at the boundaries. Note that discontinuity means we get completely different values for almost the same input. More often we would like to have continuity at the boundaries or knots: $f(\xi^+) = f(\xi^-)$ which implies the condition:$\beta_1 + \xi\beta_2 = \beta_3 + \xi_1\beta_4$ which removes one degree of freedom.

These constrains, one for each knot, can be embedded in additional basis functions of the type $h(X) = (X - \xi)_+$ where $(\cdot)_+ = \max(\cdot, 0)$ i.e., the basis function is 0 for $X - \xi < \xi$, $X - \xi$ otherwise

$$h(X) = (X - \xi)_+ = \max(X - \xi, 0) = \begin{cases} 0 & X - \xi < 0 \\ X - \xi & X - \xi \geq 0 \end{cases} \tag{6.4}$$

Using the previous and these basis functions we can reexpress the global function as:

$$\begin{aligned} f(x) =& \beta_{0,0} + \beta_{0,1}X + \cdots + \beta_{0,k-1}X^{k-1} + \\ &+ \beta_{1,0}(X - \xi_0)^0_+ + \beta_{1,1}(X - \xi_0)^1_+ + \cdots + \beta_{1,k-1}(X - \xi_0)^{k-1}_+ + \\ &+ \vdots \\ &+ \beta_{m-1,0}(X - \xi_{m-1})^0_+ + \cdots + \beta_{m-1,k-1}(X - \xi_{m-2})^{k-1}_+ \end{aligned} \tag{6.5}$$

To force continuity between two regions we set the terms $\beta_{1,0}(X - \xi_j)^0_+$ to 0:

For example forcing continuity at the first knot:

$$\begin{aligned} &\beta_{1,0}(\mathrm{X}-\xi_0)_+^0 = \beta_{0,0}+\beta_{0,1}\xi_0 \\ &\Rightarrow \beta_{1,0} = \beta_{0,0}+\beta_{0,1}\xi_0 \end{aligned} \tag{6.6}$$

In this way we don't have a jump at the boundaries:

$$\begin{aligned} \mathrm{f(x)} =&\beta_{0,0}+\beta_{0,1}\mathrm{X}+\cdots+\beta_{0,\mathrm{k-1}}\mathrm{X}^{\mathrm{k-1}}+ \\ &+\beta_{1,1}(\mathrm{X}-\xi_0)_+^1+\cdots+\beta_{1,\mathrm{k-1}}(\mathrm{X}-\xi_0)_+^{\mathrm{k-1}}+ \\ &+\vdots \\ &+\beta_{\mathrm{m-1},1}(\mathrm{X}-\xi_{\mathrm{m-1}})_+^1+\cdots+\beta_{\mathrm{m-1,k-1}}(\mathrm{X}-\xi_{\mathrm{m-1}})_+^{\mathrm{k-1}} \end{aligned} \tag{6.7}$$

In this way we have removed $\mathrm{m}-1$ parameters.

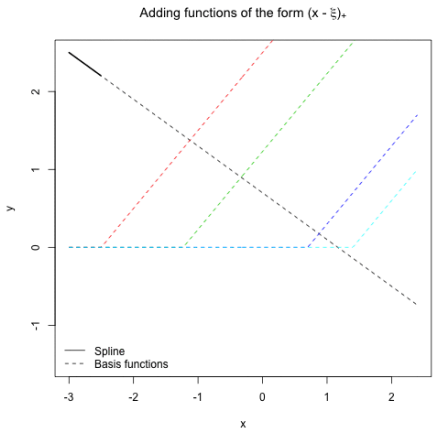
There is a difference in this representation: before we have to select a polynomial function for a specific region where the fit to the global function was given by the fit of the parameters specific to that region, now we have a global function that works a little differently. In the first region, the only non-null terms are $\beta_{0,\mathrm{j}}$ as before. When reaching the second region, the terms $\beta_{1,\mathrm{j}}$ becomes non-null. Now the coefficients of the polynomial are the sum of the coefficients of these two region: i.e., in the second region we have:

$$\begin{aligned} \mathrm{f(x)} =&\beta_{0,0}-(\beta_{1,1}+\cdots+\beta_{1,\mathrm{k-1}})\xi_0+(\beta_{0,1}+\beta_{1,1})\mathrm{X}+ \\ &+\cdots+(\beta_{0,\mathrm{k-1}}+\beta_{1,\mathrm{k-1}})\mathrm{X}^{\mathrm{k-1}} \end{aligned} \tag{6.8}$$

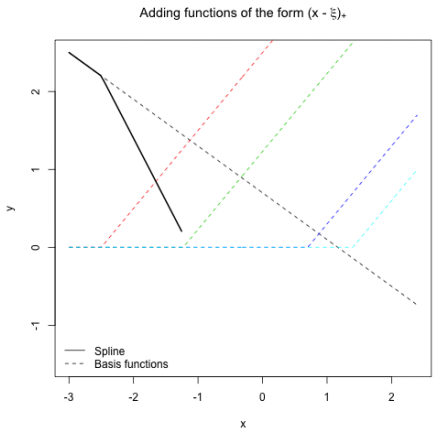
6.1 shows the global function, solid line, given by the sum of basis functions, dashed lines, when traversing the graph from left to right. At each knot, a new basis function is activated and summed to the global function.

The basis functions can be seen as a correction to the global polynomial for that specifc region and the knots are those points where the correction starts. At the knot $\mathrm{X}=\xi_{\mathrm{j}}$ so the correction starts very gently and the continuity is preserved.

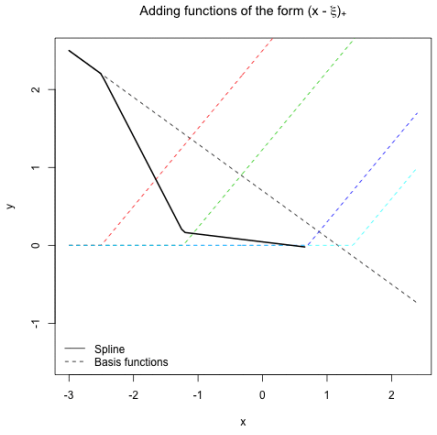
Generally, smoother functions are preferred and these are achieved by increasing the order of the local polynomials and increasing the order of continuity (continuity first derivative, second derivative). It is claimed that cubic splines are the lowest-order spline for which the discontinuity is not visible to the human eye and seldom higher degrees are needed, unless smooth derivatives are required.



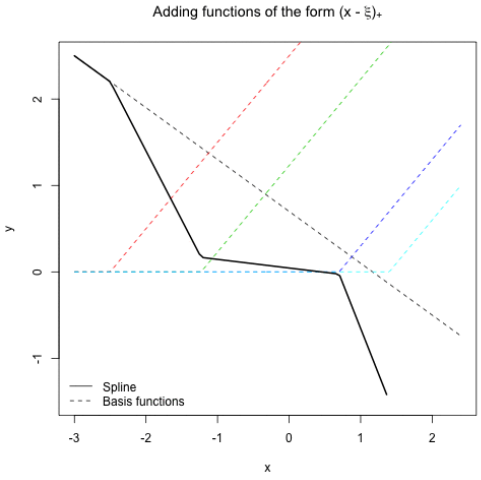
(a) First region.



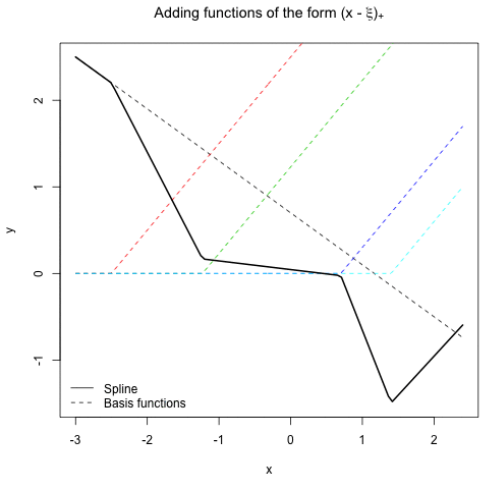
(b) Second region.



(c) Third region.



(d) Fourth region.



(e) Fifth region.

Figure 6.1: Linear splines. The solid line is the global function while the dashed lines are the basis functions.

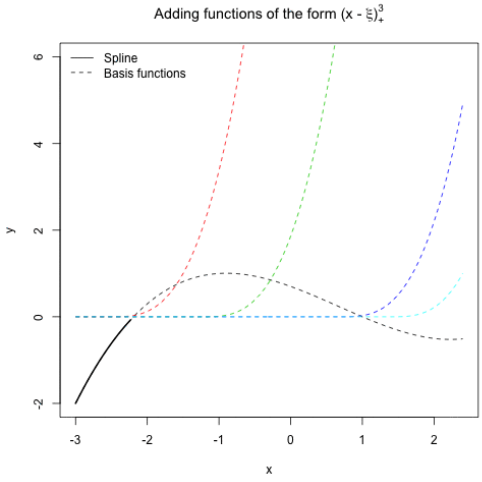
The spline order, number of knots and their positions must be selected. Generally the position of the knots is chosen according the observation, while the order is chosen a-priori.

6.2 Natural cubic splines

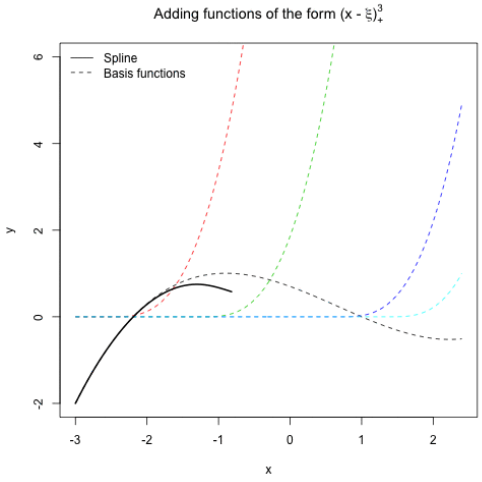
Polynomial fits to data tends to be erratic at the boundaries and additionally splines behave even more widely than the global polynomial beyond the boundaries. This is shown by the point-wise variance (the variance of the estimator at one point: $\text{Var}[\hat{f}(x_0)]$ in 6.3. Note the variance explosion close to the boundaries, especially for cubic splines.

A **natural cubic spline** adds additional constraints by forcing linearity beyond the boundary knots. These frees up four degrees of freedom (2 in both boundary regions), that can be used more profitably by sprinkling more knots in the interior region. There will be a price to pay in terms of bias near the boundaries, but in these regions we have less information anyway so assuming the function is linear here is reasonable.

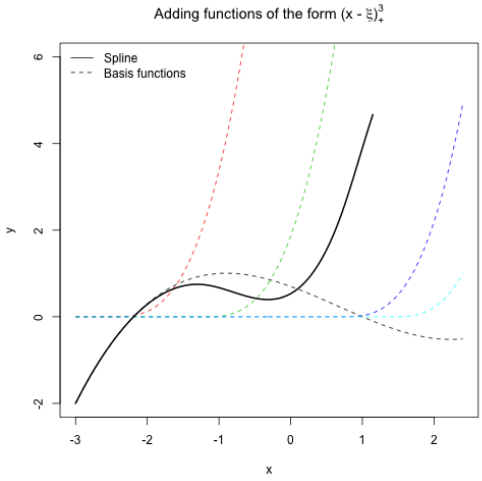
A natural cubic spline with K knots is represented by K basis functions. One can start from the basis of a cubic spline and then derive the reduce



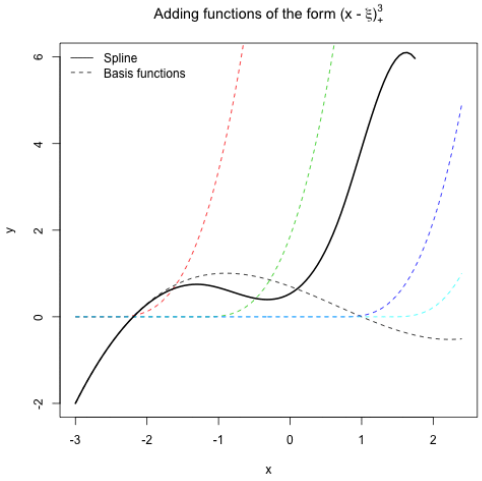
(a) First region.



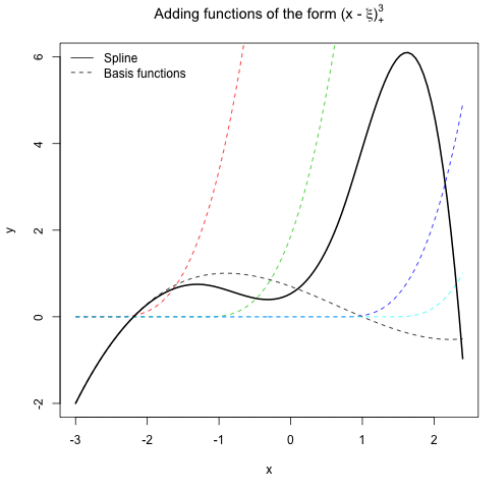
(b) Second region.



(c) Third region.



(d) Fourth region.



(e) Fifth region.

Figure 6.2: Cubic splines. The solid line is the global function while the dashed lines are the basis functions.

basis by imposing the boundary constraints.

Starting from the general continuous basis for four regions (four knots):

$$\begin{aligned} &h_1(X) = 1, \qquad h_2(X) = X, \quad h_3(X) = X^2, \quad h_4(X) = X^3, \\ &h_5(X) = X - \xi_0, \quad h_6(X) = (X - \xi_0)^2, \quad , h_7(X) = (X - \xi_0)^3 \\ &h_8(X) = X - \xi_1, \quad h_9(X) = (X - \xi_1)^2, \quad h_{10}(X) = (X - \xi_1)^3 \\ &h_9(X) = X - \xi_2, \quad h_10(X) = (X - \xi_1)^2, \quad h_{11}(X) = (X - \xi_2)^3 \end{aligned} \tag{6.9}$$

Enforcing also continuity of the first and second derivatives at the knots we reduce the basis to:

$$\begin{aligned} &h_1(X) = 1, \quad h_2(X) = X, \quad h_3(X) = X^2, \quad h_4(X) = X^3, \\ &h_5(X) = (X - \xi_0)^3, \quad h_6(X) = (X - \xi_1)^3, \quad h_7(X) = (X - \xi_2)^3. \end{aligned} \tag{6.10}$$

Enforcing linearity beyond the boundaries, we eliminate $h_3(X)$ and $h_4(X)$. We must also keep the cubic expression in the intermediate regions but somehow make the cubic part cancel out when reaching the fifth region since here it must be linear. We can add for each region in the interval $[1, K-2]$ the

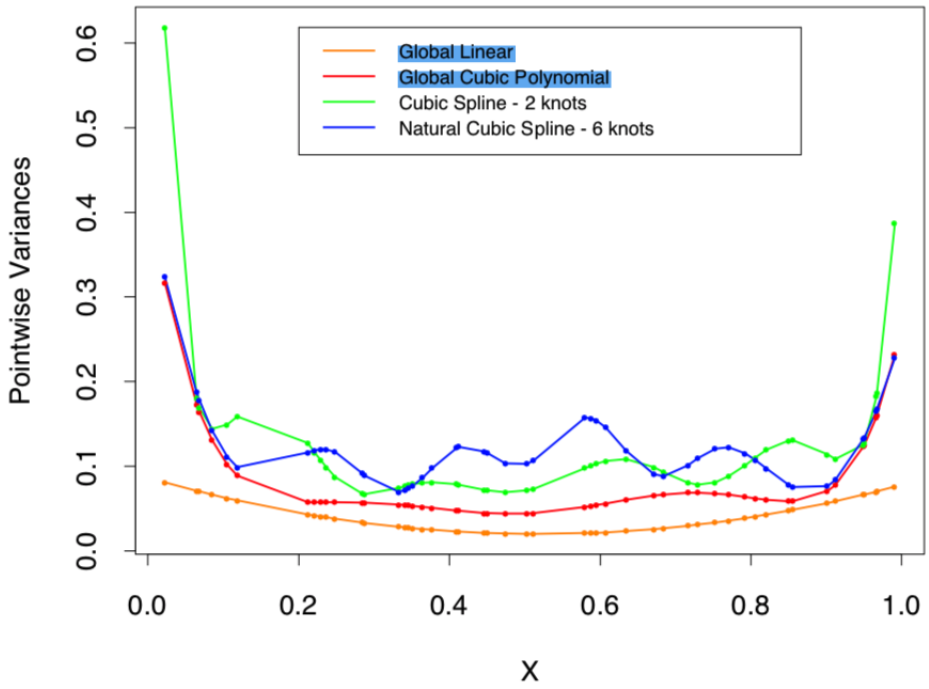


Figure 6.3: X consists of 50 points drawn randomly from U[0, 1] and an assumed error model of with constant variance. Both cubic splines have 6 degrees of freedom. The cubic spline has two knots at 0.33 and at 0.66 while the natural cubic spline has boundary knots at 0.1 and 0.9, and four interior knots uniformly spaced between them.

following basis function:

$$\begin{aligned} & h_k(X) = d_k(X) - d_{K-1}(X) \\ \textit{where} \quad & d_k(X) = \frac{(X-\xi_k)_+^3 - (X-\xi_K)_+^3}{\xi_K - \xi_k} \end{aligned} \tag{6.11}$$

In the first region we have:

$$\hat{f}_1(X) = \beta_0 + \beta_1 X \tag{6.12}$$

In the second region we have:

$$\begin{aligned} & \hat{f}_2(X) = \beta_0 + \beta_1 X + \beta_2 \frac{(X-\xi_0)^3}{\xi_2 - \xi_0} = \\ & = \beta_0 + \beta_1 X + \beta_2 \frac{X^3 - 3\xi_0 X^2 + 3\xi_0^2 X - \xi_0^3}{\xi_2 - \xi_0} = \\ & = \frac{\beta_2}{\xi_2 - \xi_0} X^3 - 3\beta_2 \frac{\xi_0}{\xi_2 - \xi_0} X^2 + (\beta_1 + 3\beta_2 \frac{\xi_0^2}{\xi_2 - \xi_0}) X + \beta_0 - \beta_2 \frac{\xi_0^3}{\xi_2 - \xi_0} \end{aligned} \tag{6.13}$$

In the third region we have:

$$
\begin{aligned}
&\hat{\mathrm{f}}_3(\mathrm{X}) = \beta_0 + \beta_1\mathrm{X} + \beta_2\mathrm{h}_2(\mathrm{X}) = \beta_0 + \beta_1\mathrm{X} + \beta_2\left[\frac{(\mathrm{X}-\xi_0)^3}{\xi_2-\xi_0} - \frac{(\mathrm{X}-\xi_1)^3}{\xi_2-\xi_1}\right] \\
&= \beta_0 + \beta_1\mathrm{X} + \beta_2\left[\frac{\mathrm{X}^3 - 3\xi_0\mathrm{X}^2 + 3\xi_0^2\mathrm{X} - \xi_0^3}{\xi_2-\xi_0} - \frac{\mathrm{X}^3 - 3\xi_1\mathrm{X}^2 + 3\xi_1^2\mathrm{X} - \xi_1^3}{\xi_2-\xi_1}\right] = \\
&= \beta_0 + \beta_1\mathrm{X} + \beta_2\frac{(\xi_2-\xi_1-\xi_2+\xi_0)\mathrm{X}^3 - 3\left[\xi_0(\xi_2-\xi_1) - \xi_1(\xi_2-\xi_0)\right]\mathrm{X}^2}{(\xi_2-\xi_0)(\xi_2-\xi_1)} + \\
&\quad + \frac{3\left[\xi_0^2(\xi_2-\xi_1) - \xi_1^2(\xi_2-\xi_0)\right]\mathrm{X} - \left[(\xi_0^3(\xi_2-\xi_1) - \xi_1^3(\xi_2-\xi_0)\right]}{(\xi_2-\xi_0)(\xi_2-\xi_1)} = \\
&= \beta_0 + \beta_1\mathrm{X} + \beta_2\frac{(-\xi_1+\xi_0)\mathrm{X}^3 - 3\xi_2(\xi_0-\xi_1)\mathrm{X}^2}{(\xi_2-\xi_0)(\xi_2-\xi_1)} + \\
&\quad + \frac{3\left[\xi_2(\xi_0-\xi_1)(\xi_0+\xi_1) - \xi_0\xi_1(\xi_0-\xi 1)\right]\mathrm{X}}{(\xi_2-\xi_0)(\xi_2-\xi_1)} + \\
&\qquad - \frac{\left[\xi_2(\xi_0-\xi_1)(\xi_0^2-\xi_0\xi_1+\xi_1^2) - \xi_0\xi_1(\xi_0-\xi 1)(\xi_0+\xi_1)\right]}{(\xi_2-\xi_0)(\xi_2-\xi_1)} = \\
&= \beta_0 + \beta_1\mathrm{X} - \beta_2(\xi_1-\xi_0)\frac{\mathrm{X}^3 - 3\xi_2\mathrm{X}^2}{(\xi_2-\xi_0)(\xi_2-\xi_1)} + \\
&\qquad + \frac{3\left[\xi_2(\xi_0+\xi_1) - \xi_0\xi_1\right]\mathrm{X} - \left[\xi_2(\xi_0^2-\xi_0\xi_1+\xi_1^2) - \xi_0\xi_1(\xi_0+\xi_1)\right]}{(\xi_2-\xi_0)(\xi_2-\xi_1)} = \\
&= -\beta_2\frac{(\xi_1-\xi_0)}{(\xi_2-\xi_0)(\xi_2-\xi_1)}\mathrm{X}^3 + 3\beta_2\frac{\xi_2(\xi_1-\xi_0)}{(\xi_2-\xi_0)(\xi_2-\xi_1)}\mathrm{X}^2 + \\
&\qquad + (\beta_1 - 3\beta_2(\xi_1-\xi_0)\frac{\xi_2(\xi_0+\xi_1) - \xi_0\xi_1}{(\xi_2-\xi_0)(\xi_2-\xi_1)})\mathrm{X} + \\
&\qquad + \beta_0 + \beta_2(\xi_1-\xi_0)\frac{\xi_2(\xi_0^2-\xi_0\xi_1+\xi_1^2) - \xi_0\xi_1(\xi_0+\xi_1)}{(\xi_2-\xi_0)(\xi_2-\xi_1)}
\end{aligned}
\tag{6.14}
$$

In the fourth region:

$$
\begin{aligned}
&\hat{f}_3(X) = \beta_0 + \beta_1 X + \beta_2 h_2(X) = \\
&= \beta_0 + \beta_1 X + \beta_2 \left[\frac{(X-\xi_0)^3 - (X-\xi_2)^3}{\xi_2 - \xi_0} - \frac{(X-\xi_1)^3 - (X-\xi_2)^3}{\xi_2 - \xi_1} \right] \\
&= \beta_0 + \beta_1 X + \beta_2 \frac{X^3 - 3\xi_0 X^2 + 3\xi_0^2 X - \xi_0^3 - X^3 + 3\xi_2 X^2 - 3\xi_2^2 X + \xi_2^3}{\xi_2 - \xi_0} + \\
&\qquad - \beta_2 \frac{X^3 - 3\xi_1 X^2 + 3\xi_1^2 X - \xi_1^3 - X^3 + 3\xi_2 X^2 - 3\xi_2^2 X + \xi_2^3}{\xi_2 - \xi_1} = \\
&= \beta_0 + \beta_1 X + \beta_2 \frac{3(\xi_2 - \xi_0) X^2 - 3(\xi_2^2 - \xi_0^2) X + \xi_2^3 - \xi_0^3}{\xi_2 - \xi_0} + \\
&\qquad - \beta_2 \frac{3(\xi_2 - \xi_1) X^2 - 3(\xi_2^2 - \xi_1^2) X + \xi_2^3 - \xi_1^3}{\xi_2 - \xi_1} = \\
&= \beta_0 + \beta_1 X + \beta_2 \frac{\cancel{3(\xi_2 - \xi_1)(\xi_2 - \xi_0) X^2} - 3(\xi_2 - \xi_1)(\xi_2^2 - \xi_0^2) X + (\xi_2 - \xi_1)(\xi_2^3 - \xi_0^3)}{(\xi_2 - \xi_0)(\xi_2 - \xi_1)} + \\
&\qquad - \beta_2 \frac{\cancel{3(\xi_2 - \xi_0)(\xi_2 - \xi_1) X^2} - 3(\xi_2 - \xi_0)(\xi_2^2 - \xi_1^2) X + (\xi_2 - \xi_0)(\xi_2^3 - \xi_1^3)}{(\xi_2 - \xi_0)(\xi_2 - \xi_1)} = \\
&= \beta_0 + \beta_1 X + \beta_2 \frac{-3(\xi_2 - \xi_1)(\xi_2^2 - \xi_0^2) X + (\xi_2 - \xi_1)(\xi_2^3 - \xi_0^3)}{(\xi_2 - \xi_0)(\xi_2 - \xi_1)} + \\
&\qquad - \beta_2 \frac{-3(\xi_2 - \xi_0)(\xi_2^2 - \xi_1^2) X + (\xi_2 - \xi_0)(\xi_2^3 - \xi_1^3)}{(\xi_2 - \xi_0)(\xi_2 - \xi_1)} = \\
&= \beta_0 + \beta_1 X + \beta_2 \frac{3\left[-\xi_2(\xi_1^2 - \xi_0^2) + \xi_2^2(\xi_1 - \xi_0) + \xi_0\xi_1(\xi_1 - \xi_0)\right] X + }{(\xi_2 - \xi_0)(\xi_2 - \xi_1)} \\
&\quad \frac{-\xi_2(\xi_1^3 - \xi_0^3) - \xi_2^3(\xi_1 - \xi_0) - \xi_0\xi_1(\xi_1 - \xi_0)}{(\xi_2 - \xi_0)(\xi_2 - \xi_1)}
\end{aligned}
\tag{6.15}
$$

Note that the fourth region is linear.

6.3 Smoothing splines

We can avoid knot selection by using a maximal set of knots, controlled by a regularization process. The problem is: *Among all functions* $f(x)$ *with two continuous derivatives, find the one that minimizes the penalized residual sum*

of squares:

$$\mathrm{RSS}(\mathrm{f},\lambda) = \sum_{\mathrm{i}=1}^{\mathrm{N}} [\mathrm{y_i} - \mathrm{f}(\mathrm{x_i})]^2 + \lambda \int \mathrm{f}''(\mathrm{t})\mathrm{dt} \tag{6.16}$$

The first term measures closeness to the data, while the second term penalizes curvature in the function. If $\lambda = 0$ f can be any function that interpolates the data so we can achieve smoothness. If $\lambda = \infty$ simple least squares line fit, since no second derivative can be tolerated and we get a "rough" function. 6.16 has an explicit finite-dimensional unique minimizer which is a natural cubic spline with knots at the unique values of the $\mathrm{x_i}$, $\mathrm{i} = 1, ..., \mathrm{N}$, so there are as many as N knots, hence N degrees of freedom. The penalty translates to a penalty on the spline coefficients, which are shrunk to the linear fit.

Being the solution a natural spline, we can write it as:

$$\mathrm{f}(\mathrm{x}) = \sum_{\mathrm{j}=1}^{\mathrm{N}} \mathrm{N_j}(\mathrm{x})\theta_\mathrm{j} \tag{6.17}$$

where $\mathrm{N_j}(\mathrm{x})$ are the N-dimensional set of basis functions for representing this family of natural splines:

$$\mathrm{RSS}(\theta,\lambda) = (\mathbf{y} - \mathbf{N}\theta)^\mathrm{T}(\mathbf{y} - \mathbf{N}\theta) + \lambda\theta^\mathrm{T}\mathbf{\Sigma_N}\theta \tag{6.18}$$

where $\mathrm{N_{i,j}} = \mathrm{N_j}(\mathrm{x_i})$ and

$$\mathbf{\Sigma_N}_{\mathrm{j,k}} = \int \mathrm{N_j''}(\mathrm{t})\mathrm{N_k''}(\mathrm{t})\mathrm{dt}$$

. The solution is :

$$\hat{\theta} = \left(\mathbf{N}^\mathrm{T}\mathbf{N} + \lambda\mathbf{\Sigma}_\mathrm{N}\right)^{-1}\mathbf{N}^\mathrm{T}\mathbf{y} = \mathbf{S}_\lambda\mathbf{y} \tag{6.19}$$

which is a generalized ridge regression. $\mathbf{S}_\lambda$ is positive-definite, symmetric and $\mathbf{S}_\lambda\mathbf{S}_\lambda \leq \mathbf{S}_\lambda$. Since it is symmetric and positive-definite it has a real eigenvalue decomposition. Let us re-write $\mathbf{S}_\lambda = (\mathbf{I} - \lambda\mathbf{K})^{-1}$. The eigenvalue decomposition of $\mathbf{S}_\lambda$ is:

$$\begin{aligned} \mathbf{S}_\lambda &= \sum_{\mathrm{k}=1}^{\mathrm{N}} \rho_\mathrm{k}(\lambda)\mathbf{u_k}\mathbf{u_k^T} \\ \rho_\mathrm{k}(\lambda) &= \frac{1}{1+\lambda\mathrm{d_k}} \end{aligned} \tag{6.20}$$

The degrees of freedom are $\mathrm{df}_\lambda = \mathrm{Tr}(\mathbf{S}_\lambda)$.

The eigenvectors are not affected by changes in λ, and hence the whole family of smoothing splines for a particular sequence of x, indexed by λ have the same eigenvectors. $\mathbf{S}_\lambda \mathbf{y} = \sum_{k=1}^{N} \mathbf{u}_k \rho_k(\lambda) \langle \mathbf{u}_k, \mathbf{y} \rangle$: the smoothing spline decomposes $\mathbf{y}$ w.r.t. the complete basis $\mathbf{u}_k$ and differently shrinking the contributions using $\rho_k(\lambda)$. So for example suppose $\mathbf{B}_x i$ is a matrix of cubic-splines basis functions. Then the vector of fitted spline values is given by $\hat{f} = \mathbf{B}_\xi (\mathbf{B}_\xi^T \mathbf{B}_\xi)^{-1} \mathbf{B}_\xi^T \mathbf{y} = \mathbf{H}_\xi \mathbf{y}$.In such a basis regression method the contributions are either left alone or shrunk to 0. The first two eigenvalues are always 1 and they correspond to the two-dimensional eigenspace of functions linear in x which are never shrunk. The eigenvalues $\rho_k(\lambda) = 1/(1 + \lambda dk)$ are an inverse function of the eigenvalues d_k of the penalty matrix $\mathbf{K}$, moderated by λ; λ controls the rate at which the $\rho_k(\lambda)$ decrease to zero. $d1 = d2 = 0$ and again linear functions are not penalized.

6.4 Multidimensional splines

Suppose $X \in \mathcal{R}^2$ and suppose we have two sets: $h_{1,k}(X_1), k = 1, \cdots, M_1$ for representing functions of coordinate X_1, and a set of M_2 functions $h_{2,k}(X_2)$ for coordinate X_2. Then the$M_1 \times M_2$ dimensional tensor product basis defined by:

$$g_{j,k} = h_{1,j}(X_1) h_{2,k}(X_2), \quad j = 1, \cdots, M_1, \quad k = 1, \cdots, M_2 \tag{6.21}$$

can be used for representing a two-dimensional function:

$$g(X) = \sum_{j=1}^{M_1} \sum_{k=1}^{M_2} \theta_{jk} g_{jk}(X) \tag{6.22}$$

The coefficients can be fit with least squares. This can be generalized to d dimensions, but note that the dimension of the basis grows exponentially fast.

One-dimensional smoothing splines (via regularization) generalize to higher dimensions as well. Suppose we have pairs y_i, x_i with $x_i \in \mathcal{R}^d$, and we seek a d-dimensional regression function f(x). The idea is to set up the problem:

$$\min_f \sum_{i=1}^{N} [y_i - f(x_i)]^2 + \lambda J[f] \tag{6.23}$$

where J is an appropriate penalty functional for stabilizing a function f in $\mathcal{R}^{\mathrm{d}}$. For example:

$$\mathrm{J}\left[\mathrm{f}\right] = \int\int_{\mathcal{R}^2} \left[\left(\frac{\partial^2 \mathrm{f}(\mathrm{x})}{\partial \mathrm{x}_1^2} \right)^2 + 2 \left(\frac{\partial^2 \mathrm{f}(\mathrm{x})}{\partial \mathrm{x}_1 \partial \mathrm{x}_2} \right)^2 + \left(\frac{\partial^2 \mathrm{f}(\mathrm{x})}{\partial \mathrm{x}_2^2} \right)^2 \right] \mathrm{dx}_1 \mathrm{dx}_2 \quad (6.24)$$

If $\lambda \to 0$ the solution approaches an interpolating function, as λ the solution approaches the least squares plane. The solution has the form:

$$\mathrm{f}(\mathrm{x}) = \beta_0 + \beta^{\mathrm{T}}\mathrm{x} + \sum_{\mathrm{j}=1}^{\mathrm{N}} \alpha_{\mathrm{j}} \mathrm{h}_{\mathrm{j}}(\mathrm{x}) \quad (6.25)$$

Additive spline model can be represented in this formulation too: there exists a penalty J[f] that guarantees that the solution has the from $\mathrm{f}(\mathrm{X}) = \alpha + \mathrm{f}_1(\mathrm{X}_1) + \cdots + \mathrm{f}_{\mathrm{d}}(\mathrm{X}_{\mathrm{d}})$ and that each of the functions f_{j} are univariate splines. In this case it is better to assume f is additive and impose an additional penalty and impose an additional penalty on each of the component functions:

$$\mathrm{J}[\mathrm{f}] = \mathrm{J}[\mathrm{f}_1 + \mathrm{f}_2 + \cdots + \mathrm{f}_{\mathrm{d}}] = \sum_{\mathrm{j}=1}^{\mathrm{d}} \int \mathrm{f}_{\mathrm{j}}''(\mathrm{t}_{\mathrm{j}}) \mathrm{dt}_{\mathrm{j}} \quad (6.26)$$

These are naturally extended to ANOVA spline decomposition:

$$\mathrm{f}(\mathrm{X}) = \alpha + \sum_{\mathrm{j}} \mathrm{f}_{\mathrm{j}}(\mathrm{X}_{\mathrm{j}}) + \sum_{\mathrm{j}<\mathrm{k}} \mathrm{f}_{\mathrm{jk}(\mathrm{X}_{\mathrm{j}}, \mathrm{X}_{\mathrm{k}}} + \cdots \quad (6.27)$$

where each of the components are splines of the required dimension. There are many choices to be made:

- maximum order of interaction (the one above is up to order 2);
- which term to include
- the representation to use (regression splines with a small number of basis functions and their tensor product for interaction or a complete basis as in smoothing splines and include appropriate regularizers for each term in the expansion.

When the number of potential dimensions is large automatic methods are desirable (for example MARS and MART procedures).

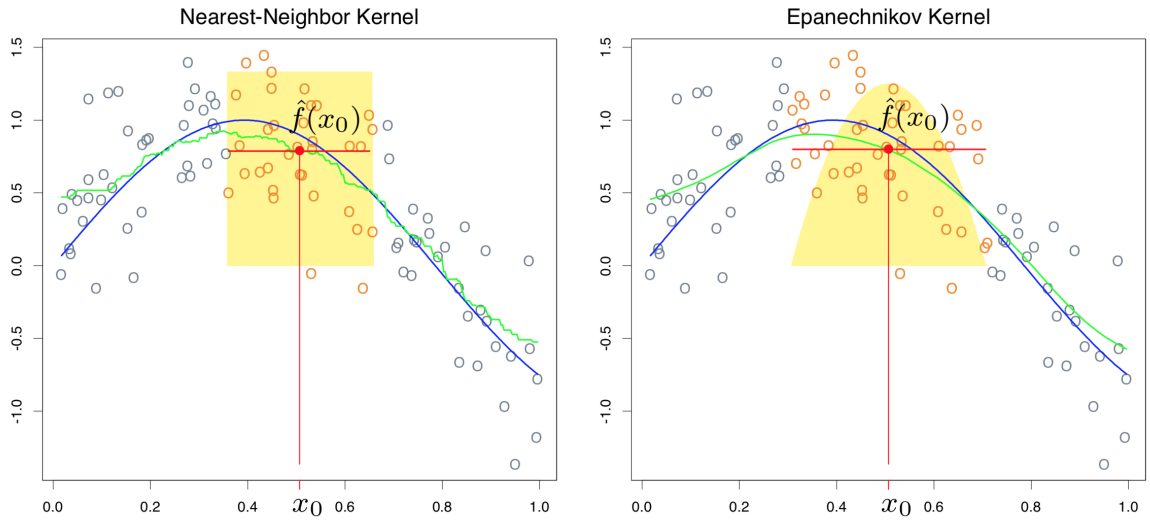


Figure 7.1: In green the fitting curves. On the left the one resulting from *k-nearest neighbourhood*, on the right the metric neighbourhood.

6.5 Wavelet smoothing

. . .

7 Kernel smoothing methods

7.1 One-dimensional kernel smoothers

We have seen the *k-nearest neighbourhood* average as an estimate of the regression function $\mathbf{E}(\mathrm{Y}|\mathrm{X}=\mathrm{x})$:

$$\hat{\mathrm{f}} = \mathrm{Ave}(\mathrm{y_i}|\mathrm{x_i} \in \mathrm{N_k}(\mathrm{x})) \tag{7.1}$$

where $\mathrm{N_k}(\mathrm{x})$ is the set of k – points nearest to x in squared distance. An example of the resulting fitting curve is shown in green on the left of 7.1. The green curve is bumpy, since $\hat{\mathrm{f}}(\mathrm{x})$ is discontinuous in x. As we move $\mathrm{x_0}$ from left to right, the k-nearest neighborhood remains constant, until a point $\mathrm{x_i}$ to the right of $\mathrm{x_0}$ becomes closer than the furthest point $\mathrm{x_{i'}}$ in the neighbourhood to the left of $\mathrm{x_0}$, at which time $\mathrm{x_i}$ replaces $\mathrm{x_{i'}}$.

Rather than give all the points in the neighbourhood equal weights, we can assign weights that die off smoothly with distance from the target

point. The right panel in 7.1 shows an example of this, using the so-called Nadaraya–Watson kernel-weighted average:

$$\hat{f}(x) = \frac{K_\lambda(x_0, x_i) y_i}{\sum_{i=1}^{N} K_\lambda(x_0, x_i)} \tag{7.2}$$

with the Epanechnikov quadratic kernel:

$$K_\lambda(x_0, x_i) = D\left(\frac{|x - x_0|}{\lambda}\right) = \begin{cases} \frac{3}{4}\left[1 - \left(\frac{|x-x_0|}{\lambda}\right)^2\right] & \text{if} \frac{|x-x_0|}{\lambda} \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{7.3}$$

Now the function is continuous. As we move the target from left to right, points enter the neighborhood initially with weight zero, and then their contribution slowly increases. Note that λ dictates the width of the window which is fixed. On the contrary, for the *k-nearest neighbourhood* the window size depends on the density since anyway k points must be considered. 7.2 shows other kernels. Another popular one is the tri-cube function:

$$D\left(\frac{|x - x_0|}{\lambda}\right) = \begin{cases} \left[1 - \left(\frac{|x-x_0|}{\lambda}\right)^3\right]^3 & \text{if} \frac{|x-x_0|}{\lambda} \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{7.4}$$

Both are compact functions. A common non-compact function is the Gaussian function.

Generalizing we can use a width function $h_\lambda(x_0)$ that determines the width of the neighbourhood at x_0 (previously we have used $h_\lambda(x_0) = \lambda$.

There are a number of details to be clarified:

- the value of λ which selects the window size. Large values imply lower variance but higher bias;
- $h_\lambda(x_0)$ can be constant (i.e., $h_\lambda(x_0) = \lambda$) or variable. Constant values (**metric window widths** keep the bias constant but the variance is inversely proportional to the local density. Nearest neighbourhood has opposite behaviour: the variance stays constant and the absolute bias varies inversely with local density.
- boundary issues: the metric neighbourhood tend to contain less points on the boundaries while the nearest-neighbourhood get wider.

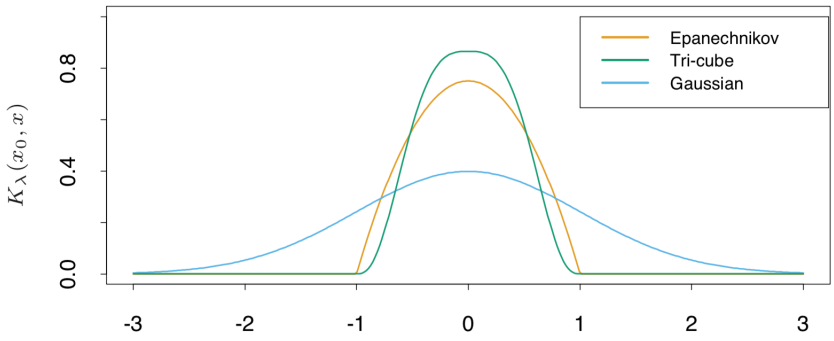


Figure 7.2: A comparison of three popular kernels for local smoothing. Each has been calibrated to integrate to 1. The tri-cube kernel is compact and has two continuous derivatives at the boundary of its support, while the Epanechnikov ker- nel has none. The Gaussian kernel is continuously differentiable, but has infinite support.

7.1.1 Local linear regression

Smooth kernel fit still has a problem close to the boundaries: because of the asymmetry in this region weighted averages can be badly biased (left in 7.3). By fitting straight lines rather than constants locally, we can remove this bias effect. Actually, this bias can be present in the interior of the domain as well, if the X values are not equally spaced. Locally weighted regression solves a separate weighted least squares problem at each target point x_0:

$$\min_{\alpha(x_0),\beta(x_0)} \sum_{i=1}^{N} K_\lambda(x_0, x_1)\,[y_i - \alpha(x_0) - \beta(x_0)x_i]^2 \tag{7.5}$$

and the estimate becomes:

$$\hat{f}(x_0) = \hat{\alpha}(x_0) + \hat{\beta}(x_0)x_0 \tag{7.6}$$

Notice that although we fit an entire linear model to the data in the region, we only use it to evaluate the fit at the single point x_0.

Define the vector-valued function $b(x)^T = (1, x)$. Let B be the $N \times 2$ regression matrix with i-th row $b(x_i)^T$, and $W(x_0)$ the $N \times N$ diagonal matrix with $i-th$ diagonal element $K_\lambda(x_0, x_i)$. Then

$$\hat{f}(x_0) = b(x_0)^T(\mathbf{B}^T\mathbf{W}(x_0)\mathbf{B})^{-1}\mathbf{B}^T\mathbf{W}(x_0)\mathbf{y} = \sum_{i=1}^{N} l_i(x_0)y_i \tag{7.7}$$

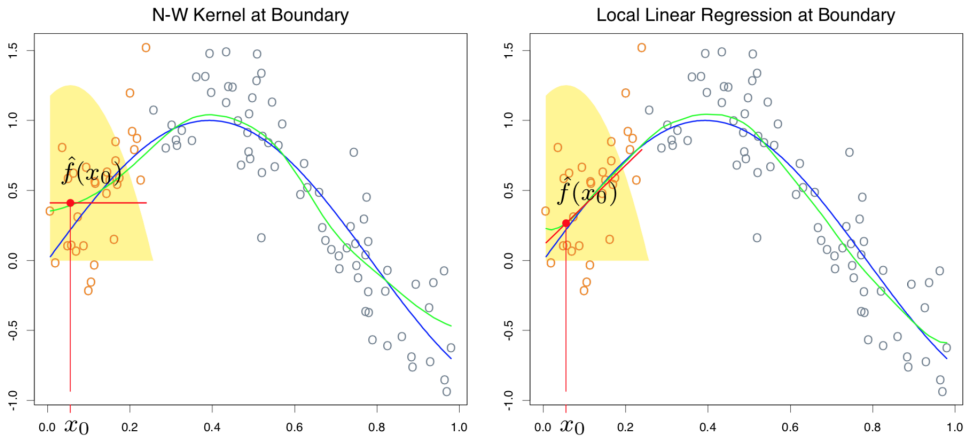


Figure 7.3: On the left it can be noticed the bias effect at the boundaries.

The weights $l_i(x_0)$ combine the weighting kernel $K_\lambda(x_0, \cdot)$ and the least squares operations and are referred to as the equivalent kernel.Local linear regression automatically modifies the kernel to correct the bias exactly to first order, a phenomenon dubbed as **automatic kernel carpentry**.

7.1.2 Local polynomial regression

We can fit local polynomial fits of any degree d:

$$\begin{aligned} &\min_{\alpha(x_0),\beta_j(x_0),j=1,\cdots,d} \sum_{i=1}^{N} K_\lambda(x_0, x_i) \left[y_i - \alpha_0 - \sum_{j=1}^{d} \beta_j(x_0) x_i^j \right]^2 \\ &\hat{f} = \hat{\alpha}(x_0) + \sum_{j=1}^{d} \hat{\beta}_j(x_0) x_0^j \end{aligned} \tag{7.8}$$

Local linear fits tend to be biased in regions of curvature of the true function, a phenomenon referred to as **trimming the hills** and **filling the valleys**. Local quadratic regression is generally able to correct this bias. There is of course a price to be paid for this bias reduction, and that is increased variance. The fit in the right panel of 7.4 is slightly more wiggly, especially in the tails. Summarizing, local linear fits can help bias dramatically at the boundaries at a modest cost in variance. Local quadratic fits do little at the boundaries for bias, but increase the variance a lot. Local quadratic fits

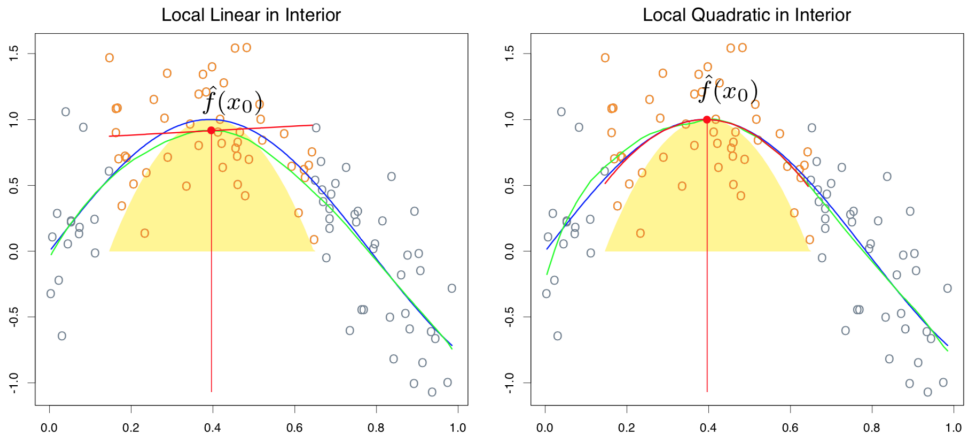


Figure 7.4: Local linear fits exhibit bias in regions of curvature of the true function. Local quadratic fits tend to eliminate this bias.

tend to be most helpful in reducing bias due to curvature in the interior of the domain. Asymptotic analysis suggest that local polynomials of odd degree dominate those of even degree. This is largely due to the fact that asymptotically the MSE is dominated by boundary effects.

7.2 Local regression in $\mathcal{R}^{\mathrm{p}}$

Let b(X) be a vector of polynomial terms in X of maximum degree d. For example, with $\mathrm{d} = 1$ and $\mathrm{p} = 2$ we get $\mathrm{b}(\mathrm{X}) = (1, \mathrm{X}_1, \mathrm{X}_2)$; with $\mathrm{d} = 2$ we get $\mathrm{b}(\mathrm{X}) = (1, \mathrm{X}_1, \mathrm{X}2, \mathrm{X}_1^2, \mathrm{X}_2^2, \mathrm{X}_1\mathrm{X}_2)$; and trivially with $\mathrm{d} = 0$ we get $\mathrm{b}(\mathrm{X}) = 1$. At each $\mathrm{x}_0 \in \mathcal{R}^{\mathrm{p}}$ solve:

$$\min_{\beta(\mathrm{x}_0)} \sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{K}_\lambda(\mathrm{x}_0, \mathrm{x}_1) \left(\mathrm{y}_\mathrm{i} - \mathrm{b}(\mathrm{x}_\mathrm{i})^{\mathrm{T}} \beta\,(\mathrm{x}_0)\right)^2 \tag{7.9}$$

to produce the fit

$$\hat{\mathrm{f}}(\mathrm{x}_0) = \mathrm{b}(\mathrm{x}_0)^{\mathrm{T}} \hat{\beta}(\mathrm{x}_0) \tag{7.10}$$

While boundary effects are a problem in one-dimensional smoothing, they are a much bigger problem in two or higher dimensions, since the fraction of points on the boundary is larger. In fact, one of the manifestations of the curse of dimensionality is that the fraction of points close to the boundary

increases to one as the dimension grows. Directly modifying the kernel to accommodate two-dimensional boundaries becomes very messy, especially for irregular boundaries. Directly modifying the kernel to accommodate two-dimensional boundaries becomes very messy, especially for irregular boundaries. Local regression becomes less useful in dimensions much larger than 2.

For high dimensionality it is impossible to simultaneously maintain localness (i.e., low bias) and a sizable sample in the neighbourhood (low variance) as dimensions increase, without the total sample size increasing exponentially with p.

7.3 Structured local regression models in $\mathcal{R}^{\mathrm{p}}$

When the dimension to sample-size ratio is unfavorable, local regression does not help us much, unless we are willing to make some structural assumptions about the model.

7.3.1 Structured kernels

The default spherical kernel gives equal weight to each coordinate and we can standardize the variables to unit deviation.

Another way is to use a positive definite matrix A to weigh the different coordinates:

$$\mathrm{K}_{\lambda}(\mathrm{x}_0, \mathrm{x}) = \mathrm{D}\left(\frac{(\mathrm{x}-\mathrm{x}_0)^{\mathrm{T}}\mathbf{A}(\mathrm{x}-\mathrm{x}_0)}{\lambda}\right) \tag{7.11}$$

Entire coordinates or directions can be downgraded or omitted by imposing appropriate restrictions on $\mathbf{A}$. For example, if $\mathbf{A}$ is diagonal, then we can increase or decrease the influence of individual predictors X_{j} by increasing or decreasing A_{jj}.

7.3.2 Structured regression function

We are trying to fit a regression function $\mathbf{E}[\mathrm{Y}|\mathrm{X}] = \mathrm{f}(\mathrm{X}_1, \mathrm{X}_2, \cdots, \mathrm{X}_{\mathrm{p}})$ in $\mathrm{CMcalR}^{\mathrm{p}}$ in which every level of interaction is potentially present. It is natural to consider analysis-of-variance (ANOVA) decompositions of the form

$$\mathrm{f}(\mathrm{X}_1, \mathrm{X}_2, \cdots, \mathrm{X}_{\mathrm{p}}) = \alpha + \sum_{\mathrm{j}} \mathrm{g}_{\mathrm{j}}(\mathrm{X}_{\mathrm{j}}) + \sum_{\mathrm{k}<\mathrm{l}} \mathrm{g}_{\mathrm{kl}}(\mathrm{X}_{\mathrm{k}}, \mathrm{X}_{\mathrm{l}}) + \cdots \tag{7.12}$$

and then introduce structure by eliminating the higher-order terms. Additive models assume only main effect terms: $f(X) = \alpha + \sum_{j=1}^{p} g_j(X_j)$, second order models will have terms with interactions of order at most two and so on. The important detail is that at any stage, one-dimensional local regression is all that is needed. The same ideas can be used to fit low-dimensional ANOVA decompositions.

7.4 Local likelihood and other models

Any parametric model can be made local.

For example we can use likelihood local to x_0, with the parameters $\theta(x_0) = x_0^T \beta(x_0)$:

$$l(\beta(x_0)) = \sum_{i=1}^{N} K_\lambda(x_0, x_i) l\Big(y_i, x_i^T \beta(x_0)\Big) \tag{7.13}$$

Also time-series can be made local by fitting by local least squares with a kernel in order to vary according to the short-term history of the series.

7.4.1 Local multiclassifier linear logistic regression

Consider categorical responses $g_i \in 1, 2, \cdots, J$. The global linear model is:

$$\Pr(G = j | X = x) = \frac{e^{\beta_{j0} + \beta_j^T x}}{1 + \sum_{k=1}^{J-1} e^{\beta_{k0} + \beta_k^T x}}$$

The local log-likelihood for the class J can be written as:

$$\begin{aligned} &\sum_{i=1}^{N} K_\lambda(x_0, x_i) \Bigg\{ \beta_{g_i 0}(x_0) + \beta_{g_i}(x_0)^T (x_i - x_0) + \\ &- \log \left[1 + \sum_{k=1}^{J-1} e^{\beta_{k0}(x_0) + \beta_k(x_0)^T (x_i - x_0)} \right] \Bigg\} \end{aligned} \tag{7.14}$$

Notice that $\beta_{J0} = \beta_J = 0$ i.e., the last class is taken as reference. Also, we have centred the local regression at x_0, so that the fitted posterior probabilities at x_0 are simply:

$$\hat{\Pr}(G = j, X = x_0) = \frac{e^{\hat{\beta}_{j0}(x_0)}}{1 + \sum_{k=1}^{J-1} e^{\hat{\beta}_{k0}(x_0)}} \tag{7.15}$$

7.5 Kernel density classification

One can use nonparametric density estimates for classification in a straightforward fashion using Bayes' theorem:

$$\hat{\Pr}(G = k|X = x_0) = \frac{\hat{\pi}_k \hat{f}_k(x_0)}{\sum_{j=1}^{J} \hat{\pi}_k \hat{f}_j(x_0)} \tag{7.16}$$

with $\hat{f}_j$ being a fitted nonparametric density estimates for each class separately.

7.6 Naive-Bayes classifier

It assumes that given a class $G = j$, the features X_k are independent:

$$f_j(X) = \prod_{k=1}^{p} f_{jk}(X_k) \tag{7.17}$$

While this assumption is generally not true, it does simplify the estimation dramatically since f_{jk} can be estimated separately using one dimensional kernel density estimates. Despite these rather optimistic assumptions, Naive-Bayes classifiers often outperform far more sophisticated alternatives. The reasons are that although the individual class density estimates may be biased, this bias might not hurt the posterior probabilities as much, especially near the decision regions. In fact, the problem may be able to withstand considerable bias for the savings in variance such a "naive" assumption earns.

7.7 Radial basis functions

Radial basis functions combine the idea of basis methods and kernel functions by treating $K_\lambda(\xi, x)$ as a basis function:

$$f(x) = \sum_{j=1}^{M} K_{\lambda_j}(\xi_j, x)\beta_j = \sum_{j=1}^{M} D\left(\frac{\|x - \xi_j\|}{\lambda_j}\right)\beta_j \tag{7.18}$$

A popular choice for D is the standard Gaussian density function. There are several methods to estimate the parameters, among which one of the

most common is the least squares. We can optimize the sum-of-squares with respect to all parameters:

$$\min_{\{\lambda_j,\xi_j,\beta_j\}_1^M} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{M} \beta_j e^{-\frac{(x_i-\xi_j)^T(x_i-\xi_j)}{\lambda_j^2}} \right)^2 \tag{7.19}$$

This model is referred to as RBF network, with ξ_J and λ_j playing the role of weights. This criterion is non-convex with multiple local minima.

Another way is to estimate $\{\lambda_j, \xi_j\}$ separately from β_j. Often, the kernel parameters λ_j and ξ_j are chosen in an unsupervised way using the X distribution alone. One of the method is to fit a Gaussian mixture density model to the training x_i, which provides both the centres ξ_j and the scales λ_j. Other approaches use cluster methods to locate ξ_j and treat $\lambda_j = \lambda$ as a hyper-parameter.

Using a constant λ might create holes-regions (regions that are not fitted by any gaussian) where no kernels has appreciable support. Renormalized radial basis functions solve this problem:

$$h_j(x) = \frac{D\left(\frac{\|x-\xi_j\|}{\lambda}\right)}{\sum_{k=1}^{M} D\left(\frac{\|x-\xi_k\|}{\lambda}\right)} \tag{7.20}$$

7.8 Mixture Models for Density Estimation and Classification

The mixture model can be viewed as a kind of kernel method:

$$\begin{aligned} f(x) &= \sum_{m=1}^{M} \alpha_m \phi(x, \mu_m, \Sigma_m) \\ \sum_m \alpha_m &= 1 \end{aligned} \tag{7.21}$$

It is possible to use any density instead of the Gaussian densities.

The parameters are usually fit by maximum likelihood, using the EM algorithm. If the covariance matrix is force to be scalar $\Sigma_m = \sigma_m \mathbf{I}$, then 7.21 has the form of a radial basis expansion. If $\sigma_m = \sigma > 0$ and $M \uparrow$

8 Model assessment and selection

Take a look also at this: https://machinelearningmastery.com/probabilistic-model-selection-measures/

Definition 8.1. Training error: it is the average loss over the training samples:

$$\bar{\text{err}} = \frac{1}{\text{N}} \sum_{\text{i}=1}^{\text{N}} \text{L}(\text{y}_\text{i}, \hat{\text{f}}(\text{x}_\text{i})) \tag{8.1}$$

For categorical variables it is:

$$\bar{\text{err}} = -\frac{2}{\text{N}} \sum_{\text{i}=1}^{\text{N}} \log \hat{\text{p}}_\text{k}(\text{x}_\text{i}) \tag{8.2}$$

The "–2" in the definition makes the log-likelihood loss for the Gaussian distribution match squared-error loss.

Definition 8.2. Test error: with the term *test error* or *generalization error* we indicate the prediction error over an independent test sample different from the one used for the training and it is indicated as

$$\text{Err}_\tau \left[\text{L}(\text{Y}, \hat{\text{f}}(\text{X}) | \tau \right] \tag{8.3}$$

where τ indicates the training set and X and Y are drawn randomly from their joint distribution.

Definition 8.3. Expected prediction error: or **expected test error** is the expectation of the test.

$$\text{Err} = \mathbf{E} \left[\text{L}(\text{Y}, \hat{\text{f}}(\text{X})) \right] = \mathbf{E} \left[\text{Err}_\tau \right] \tag{8.4}$$

8.1 shows the prediction errors of the same type of model trained with different training sets in light red and their expectation (average) in dark red.

To judge the prediction capability of a model we need to estimate Err_τ. As the model becomes more and more complex, it uses the training data more and is able to adapt to more complicated underlying structures. Hence there is a decrease in bias but an increase in variance. There is some intermediate

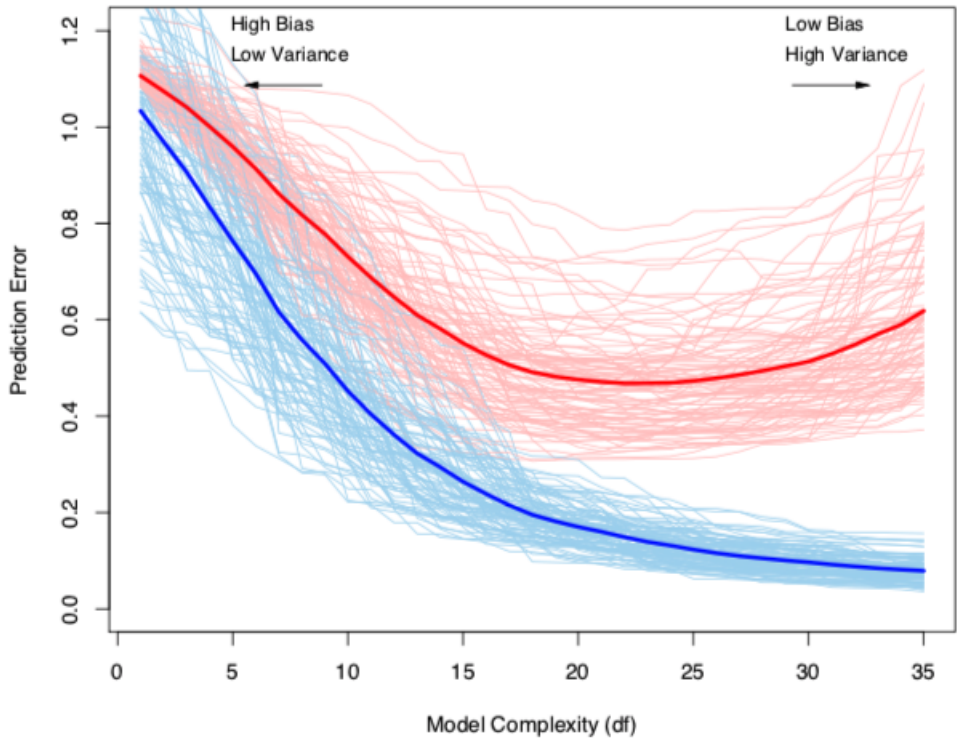


Figure 8.1: Behaviour of test sample and training sample error as the model complexity is varied. The light blue curves show the training error $\bar{\text{err}}$, while the light red curves show the conditional test error $\text{Err}_{\mathcal{T}}$ for 100 training sets of size 50 each, as the model complexity is increased. The solid curves show the expected test error Err and the expected training error $\text{E}[\bar{\text{err}}]$.

model complexity that gives minimum expected test error. The training error is not reliable because it always decreases with the model complexity and with too much complexity it looses the capability to generalize.

The same holds for categorical variables (introduced in 8.1). Typical loss functions are:

$$\begin{aligned} &\mathrm{L}(\mathrm{G}, \hat{\mathrm{G}}) = \mathrm{I}(\mathrm{G} \neq \hat{\mathrm{G}}(\mathrm{X})) \qquad \textit{0-1 loss}, \\ &\mathrm{L}(\mathrm{G}, \hat{\mathrm{p}}(\mathrm{X})) = -2 \sum_{\mathrm{k}=1}^{\mathrm{K}} \mathrm{I}(\mathrm{G} = \mathrm{k}) \log \hat{\mathrm{p}}_{\mathrm{k}}(\mathrm{X}) \qquad \textit{-2} \times \textit{ log-likelihood}, \end{aligned} \tag{8.5}$$

where the quantity $-2 \times \log$ –likelihood is generally referred as deviance. The "–2", as already said, in the definition makes the log-likelihood loss for the Gaussian distribution match squared-error loss. A number of methods for estimating the expected test error for a model exists. Typically our model will have a tuning parameter or parameters α and so we can write our predictions as $\hat{\mathrm{f}}(\mathrm{x})_\alpha$ The tuning parameter varies the complexity of our model, and we wish to find the value of α that minimizes error, that is, produces the minimum of the average test error curve. There are two goals that one might want to achieve:

Definition 8.4. model selection: estimating the performance of different models in order to choose the best one.

Definition 8.5. model assessment: having chosen a final model, estimate its prediction error on new data.

If we are in a data-rich situation, the best approach for both problems is to randomly divide the dataset into three parts: a training set, a validation set, and a test set. The training set is used to fit the models; the validation set is used to estimate prediction error for model selection; the test set is used for assessment of the generalization error of the final chosen model. Ideally, the test set should be kept in a "vault", and be brought out only at the end of the data analysis. It is difficult to give a general rule on how to choose the number of observations in each of the three parts, as this depends on the signal-to- noise ratio in the data and the training sample size. A typical split might be 50% for training, and 25% each for validation and testing.

The following methods allows to work in a situation with insufficient data. The methods can be distinguished in analytical (AIC, BIC, MDL, SRM) or by efficiently sample re-use (cross-validation and boot-strap). Typically the

more complex we make the model f, the lower the (squared) bias but the higher the variance.

8.1 Optimism of the training error rate

Given a training set τ the generalization error of a model is:

$$\mathrm{Err}_\tau = \mathbf{E}_{\mathrm{X}^0,\mathrm{Y}^0}\left[\mathrm{L}(\mathrm{Y}^0, \hat{\mathrm{f}}(\mathrm{X}^0))|\mathrm{tau}\right] \tag{8.6}$$

where the point $(\mathrm{X}^0, \mathrm{Y}^0)$ is a new test data point. Averaging over training sets τ (i.e., averaging over the same type of models $\hat{\mathrm{f}}$ but trained with different trainsets belonging to the same distribution):

$$\mathrm{Err} = \mathbf{E}_\tau \mathbf{E}_{\mathrm{X}^0,\mathrm{Y}^0}\left[\mathrm{L}(\mathrm{Y}^0, \hat{\mathrm{f}}(\mathrm{X}^0))|\mathrm{tau}\right] \tag{8.7}$$

The training error $\bar{\mathrm{err}} = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{L}(\mathrm{y_i}, \hat{\mathrm{f}}(\mathrm{x_i}))$ will be less than the true error, because the same data is being used to fit the method and assess its error. A fitting method typically adapts to the training data, and hence the apparent or training error $\bar{\mathrm{err}}$ will be an overly optimistic estimate of the generalization error Err_τ, which can be thought as *extra-sample error*.

The nature of the optimism in the training error can be explained considering the *in-sample error*:

$$\mathrm{Err}_{\mathrm{in}} = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{E}_{\mathrm{Y}^0}\left[\mathrm{L}(\mathrm{Y}^0_{\mathrm{i}}, \hat{\mathrm{f}}(\mathrm{x_i}))|\tau\right] \tag{8.8}$$

To explain it better consider the same input points $\mathrm{x_i}, \quad \mathrm{i} = 1, \cdots, \mathrm{N}$. In the training set for these points we got a given set of values, while on new data we get a slightly different values Y^0 due to the random process involved (for example the noise considering the additive model). The model $\hat{\mathrm{f}}$ have been trained using the training set, so for its prediction for those points will be close to the values seen in the training set (influenced by randomness) and it cannot foresee the non-deterministic part in the new samples. We define the optimism as the difference between the real $\mathrm{Err}_{\mathrm{in}}$ and the training error:

$$\mathrm{op} \equiv \mathrm{Err_in} - \bar{\mathrm{err}} \tag{8.9}$$

and this is typically positive since the second term is generally biased downward. Averaging the optimism:

$$\omega = \mathbf{E}_{\mathrm{y}}(\mathrm{op}) \tag{8.10}$$

where the notation $\mathbf{E}_{\mathrm{y}}$ has been used instead of $\mathbf{E}_{\tau}$ because the predictors (the inputs) are fixed, while the outputs vary.

For squared error, 0-1 loss functions and other loss functions one can show:

$$\omega = \frac{2}{\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{Cov}(\hat{\mathrm{y}}_{\mathrm{i}}, \mathrm{y}_{\mathrm{i}}) \tag{8.11}$$

Thus the amount by which err underestimates the true error depends on how strongly y_{i} affects its own prediction. The harder we fit the data, the greater the covariance will be, thereby increasing the optimism. We have the relationship:

$$\mathbf{E}_{\mathrm{y}}(\mathrm{Err}_{\mathrm{in}}) = \mathbf{E}_{\mathrm{y}}(\bar{\mathrm{err}}) + \frac{2}{\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{Cov}(\hat{\mathrm{y}}_{\mathrm{i}}, \mathrm{y}_{\mathrm{i}}) \tag{8.12}$$

This expression simplifies if $\hat{\mathrm{y}}_{\mathrm{i}}$ is obtained by a linear fit with d inputs or basis functions. For example for the additive model $\mathrm{Y} = \mathrm{f}(\mathrm{X}) + \epsilon$:

$$\sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{Cov}(\hat{\mathrm{y}}_{\mathrm{i}}, \mathrm{y}_{\mathrm{i}}) = \mathrm{d}\sigma_{\epsilon}^2 \tag{8.13}$$

and so

$$\mathbf{E}_{\mathrm{y}}(\mathrm{Err}_{\mathrm{in}}) = \mathbf{E}_{\mathrm{y}}(\bar{\mathrm{err}}) + \frac{2\mathrm{d}}{\mathrm{N}}\sigma_{\epsilon}^2 \tag{8.14}$$

We can use these expressions to select the effective number of parameters. The optimism increases with the number d of inputs or basis functions we use, but decreases as the training sample size increases

An obvious way to estimate prediction error is to estimate the optimism and then add it to the training error. The methods such as Cp, AIC, BIC and others work in this way, for a special class of estimates that are linear in their parameters. In contrast, cross-validation and bootstrap methods are direct estimates of the extra-sample error. These general tools can be used with any loss function, and with nonlinear, adaptive fitting techniques. In-sample error is not usually of direct interest since future values of the features

are not likely to coincide with their training set values. But for comparison between models, in-sample error is convenient and often leads to effective model selection. The reason is that the relative (rather than absolute) size of the error is what matters.

8.2 C_p metric

One can estimate the in-sample error as

$$\hat{Err} = \bar{err} + \hat{\omega} \tag{8.15}$$

where $\hat{\omega}$ is an estimate of the average optimism. Using 8.14:

$$C_p = \bar{err} + \frac{2d}{N}\hat{\sigma}_\epsilon^2 \tag{8.16}$$

where $\hat{\sigma}_\epsilon^2$ is an estimate of the noise variance.

The model with lowest C_p should be selected.

8.3 Akaike Information Criterion

Suppose we have models $\mathcal{M}_1, \cdots, \mathcal{M}_k$ where each model is a set of densities:

$$\mathcal{M}_j = \left\{ p(y; \theta_j) : \theta_j \in \Theta_j \right\}. \tag{8.17}$$

and we have data $Y_1, \cdots, Y_n$ drawn from an unknown density f that we do not know if it is in any of the models. Let $\hat{\theta}_j$ be the Maximum Likelihood estimation from the model j. Using this we can estimate f with $\hat{p}_j(y) = p(y, \hat{\theta}_j)$. The quality of $\hat{p}_j(y)$ (i.e., its distance form f) can be measured by the **Kullback-Leibler distance**:

$$\begin{aligned} K(p, \hat{p}_j) &= \int p(y) \log \frac{p(y)}{\hat{p}_j(y)} dy = \\ &= \int p(y) \log p(y) dy - \int p(y) \log \hat{p}_j(y) dy \end{aligned} \tag{8.18}$$

We want to minimize this distance. The first term is constant while we can act on $\hat{p}_j(y)$. Since it is a probability (i.e. $\hat{p}_j(y)$ is at most one, minimizing the distance is equivalent to maximizing the second term:

$$K_j = \int p(y) \log \hat{p}_j(y) dy \tag{8.19}$$

Intuitively one estimate of K_j might be:

$$\bar{K}_j = \frac{1}{N}\sum_{i=1}^{N} \log p_j(Y_i, \hat{\theta}_j) = \frac{\ell_j(\hat{\theta}_j)}{N} \tag{8.20}$$

i.e., assuming a uniform distribution. This estimation is very biased because data are being used twice: once for the MLE and the second one to calculate the integral. So we have $\bar{K}_j \neq K_j$. We want to find such difference.

Let us drop the index j (i.e., we are considering just one model and not a set) and just use K and $\bar{K}$. Let θ_0 be the parameters that minimize $K(f, p(\cdot, \theta))$. So $p(y, \theta_0)$ is the closest density in the model to the true density. Let $\ell(y, \theta) = \log p(y, \theta)$ and let us define the **score** as:

$$s(y, \theta) = \frac{\partial \log p(y, \theta)}{\partial \theta} \tag{8.21}$$

and let $H(y, \theta)$ be the matrix of second derivatives. Let

$$Z_N = \sqrt{N}\Big(\hat{\theta} - \theta_0\Big) \tag{8.22}$$

and recall (?) that :

$$Z_N \to \mathcal{N}(0, J^{-1}VJ^{-1}) \tag{8.23}$$

where

$$\begin{aligned} J &= -\mathbf{E}\left[H(Y, \theta_0\right] \\ V &= \mathrm{Var}(s(Y, \theta_0)) \end{aligned} \tag{8.24}$$

Let

$$S_N = \frac{1}{N}\sum_{i=1}^{N} s(Y_i, \theta_0) \tag{8.25}$$

By the **Central Limit Theorem**, and recalling that θ_0 is the value that minimizes K (if it is a convex function then for this value the score is 0 i.e., 0 mean):

$$\sqrt{N}S_N \to \mathcal{N}(0, V) \tag{8.26}$$

since

$$\begin{aligned}\mathrm{Var}\left[\sqrt{\mathrm{N}}\mathrm{S_n}\right] &= \mathbf{E}\left[\left(\sqrt{\mathrm{N}}\mathrm{S_n} - \cancel{\mu_{\mathrm{S_n}}}\right)^2\right] = \\ &= \mathrm{N}\mathbf{E}\left[\left(\frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}}\mathrm{s}(\mathrm{Y_i},\theta_0)\right)^2\right] = \frac{1}{\mathrm{N}}\mathbf{E}\left[\left(\sum_{\mathrm{i}=1}^{\mathrm{N}}\mathrm{s}(\mathrm{Y_i},\theta_0)\right)^2\right] = \\ &= \frac{1}{\mathrm{N}}\left(\sum_{\mathrm{i}=1}^{\mathrm{N}}\mathbf{E}\left[\mathrm{s}(\mathrm{Y_i},\theta_0)\right]^2\right) = \frac{1}{\mathrm{N}}\mathrm{NV} = \mathrm{V}\end{aligned} \tag{8.27}$$

Now J is a scalar, considering the distribution $\mathrm{JZ_N}$ where $\mathrm{Z_N} \to \mathcal{N}(0, \mathrm{J^{-1}VJ^{-1}})$, the result is still a distribution with 0. For the variance let us use the property:

$$\begin{aligned}&\mathrm{Var}\left[\mathrm{JZ_N}\right] = \mathrm{J}(\mathrm{J^{-1}VJ^{-1}})\mathrm{J^T} = \mathrm{V} \\ &\Rightarrow \mathrm{JZ_N} \sim \mathcal{N}(0, \mathrm{V})\end{aligned} \tag{8.28}$$

By using **Taylor series expansion**:

$$\mathrm{K} \approx \int \mathrm{p(y)}\left(\log \mathrm{p}(\mathrm{y},\theta_0) + \cancel{\left(\hat{\theta}-\theta_0\right)^{\mathrm{T}}\mathrm{s}(\mathrm{y},\theta_0)} + \frac{1}{2}\left(\hat{\theta}-\theta_0\right)^{\mathrm{T}}\mathrm{H}(\mathrm{y},\theta_0)\left(\hat{\theta}-\theta_0\right)\right)\mathrm{dy} \tag{8.29}$$

The second term $\left(\hat{\theta}-\theta_0\right)^{\mathrm{T}}\mathrm{s}(\mathrm{y},\theta_0)$ is 0 because the score has 0 mean. Defining

$$\mathrm{K_0} = \int \mathrm{p(y)}\log \mathrm{p}(\mathrm{y},\theta_0)\mathrm{dy} \tag{8.30}$$

we have:

$$\begin{aligned}\mathrm{K} &\approx \mathrm{K_0} + \frac{1}{\sqrt{\mathrm{N}}}\mathrm{Z_N^T}\int \mathrm{H}(\mathrm{y},\theta_0)\mathrm{p(y)dy}\frac{1}{\sqrt{\mathrm{N}}}\mathrm{Z_N} = \\ &= \mathrm{K_0} + \frac{1}{\sqrt{\mathrm{N}}}\mathrm{Z_N^T}\int \mathrm{H}(\mathrm{y},\theta_0)\mathrm{p(y)dy}\frac{1}{\sqrt{\mathrm{N}}}\mathrm{Z_N} = \mathrm{k_0} - \frac{1}{2\mathrm{N}}\mathrm{Z_N^T}\mathrm{JZ_N}\end{aligned} \tag{8.31}$$

Now applying the **Taylor series expansion** to hatK:

$$\bar{\mathrm{K}} \approx \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}}\left(\ell(\mathrm{Y_i},\theta_0) + \left(\hat{\theta}-\theta_0\right)^{\mathrm{T}}\mathrm{s}(\mathrm{Y_i},\theta_0)\right) + \frac{1}{2}\left(\hat{\theta}-\theta_0\right)^{\mathrm{T}}\mathrm{H}(\mathrm{Y_i},\theta_0)\left(\hat{\theta}-\theta_0\right) \tag{8.32}$$

Defining

$$\begin{aligned} A_N &= \frac{1}{N}\sum_{i=1}^{N} \ell(Y_i, \theta_0) - K_0 = \frac{1}{N}\sum_{i=1}^{N} (\ell(Y_i, \theta_0) - K_0) \\ J_N &= -\frac{1}{N}\sum_{i=1}^{N} H(Y_i, \theta_0) - \xrightarrow{P} J \end{aligned} \tag{8.33}$$

we have

$$\begin{aligned} \bar{K} &\approx K_0 + A_N + (\hat{\theta} - \theta_0) S_N - \frac{1}{2N} Z_N^T J_N Z_N \approx \\ &\approx K_0 + A_N + \frac{Z_N^T}{N} \sqrt{N} J^{-1} J S_N - \frac{1}{2N} Z_N^T J Z_N = \\ &= K_0 + A_N + \frac{Z_N^T J Z_N}{N} - \frac{1}{2N} Z_N^T J Z_N \end{aligned} \tag{8.34}$$

where 8.28 has been used again ($J^{-1} S_N \to J^{-1} \mathcal{N}(0, V) = \mathcal{N}(0, J^{-1} V J^{-1}) = Z_N$). We need another property. Let ϵ be a random vector with mean μ and variance Σ and let

$$Q = \epsilon^T A \epsilon \tag{8.35}$$

with Q called a quadratic form. Then

$$\mathbf{E}[Q] = \text{trace}(A\Sigma) + \mu^T A \mu \tag{8.36}$$

Now taking the difference:

$$\begin{aligned} \mathbf{E}[\text{hat}K - K] = \mathbf{E}[A_N] + \mathbf{E}\left[\frac{Z_N^T J Z_N}{N}\right] = 0 + \frac{\text{trace}(J J^{-1} V J^{-1})}{N} = \\ \frac{\text{trace}(J^{-1} V J^{-1} J)}{N} = \frac{\text{trace}(J^{-1} V)}{N} \end{aligned} \tag{8.37}$$

If the model is correct then $J^{-1} = V$ so that $\text{trace}(J^{-1} V) = \text{trace}(I) = d$:

$$K \approx \hat{K} = \bar{K} - \frac{d}{N} \tag{8.38}$$

So the bias is approximately $\frac{d}{N}$. Considering the model j:

$$\begin{aligned} \hat{K}_j &= \bar{K}_j - \frac{d_j}{N} \\ \text{AIC}(j) &= 2N\hat{K}_j = \ell_j(\hat{\theta}_j) - 2d_j \end{aligned} \tag{8.39}$$

Maximizing $\hat{\mathrm{K}}_{\mathrm{j}}$ is the same as maximising AIC(j). Constants are not important in the maximization process. With this representation we choose the model with the highest AIC. Other books use the following representation:

$$\mathrm{AIC(j)} = -\frac{2}{\mathrm{N}}\ell_{\mathrm{j}}(\hat{\theta}_{\mathrm{j}}) + 2\frac{\mathrm{d_j}}{\mathrm{N}} \tag{8.40}$$

and with this representation the best model is the one that minimizes the AIC, since the signs have been inverted.

For the Gaussian model, with variance $\sigma_\epsilon^2 = \hat{\sigma}_\epsilon^2$, the AIC statistic is equivalent to $\mathrm{C_p}$.

For non-linear and other complex models we replace d with a measure of the model complexity. Given a set of models f_α, $\bar{\mathrm{err}}(\alpha)$ is the training error and $\mathrm{d}(\alpha)$ the number of parameters:

$$\mathrm{AIC}(\alpha) = \bar{\mathrm{err}}(\alpha) + 2\frac{\mathrm{d}\alpha}{\mathrm{N}}\hat{\sigma}_\epsilon^2. \tag{8.41}$$

8.4 Effective number of parameters

A linear fitting method (linear regression, derived basis sets and smoothing methods with quadratic shrinkage such as ridge regression and cubic smoothing splines) have the form:

$$\hat{\mathbf{y}} = \mathbf{S}\mathbf{y} \tag{8.42}$$

where $\mathbf{S}$ is a $\mathrm{N} \times \mathrm{N}$ matrix. The effective number of parameters is defined as:

$$\mathrm{df}(\mathbf{S}) = \mathrm{trace}(\mathbf{S}) = \mathrm{M} \tag{8.43}$$

since $\mathbf{S}$ is an orthogonal projection matrix onto a basis set spanned by M features. Then instead of d we use trace($\mathbf{S}$).

For models like NN in which we minimize an error function with weight decay penalty (regularization) $\alpha \sum_{\mathrm{m}} \mathrm{w}_{\mathrm{m}}^2$, the effective number of parameters has ethe form:

$$\mathrm{df}(\alpha) = \sum_{\mathrm{m}=1}^{\mathrm{M}} \frac{\theta_{\mathrm{m}}}{\theta_{\mathrm{m}} + \alpha} \tag{8.44}$$

where θ_{m} are the eigenvalues of the Hessian matrix $\partial \mathrm{R}(\omega)/\partial\omega\partial\omega^{\mathrm{T}}$.

8.5 Bayesian Information Criterion

Known also as *Schwarz Criterion*, it is equivalent to the MDL. We choose j to maximize

$$\mathrm{BIC(j)} = \ell_{\mathrm{j}}(\hat{\theta}_{\mathrm{j}}) - \frac{\mathrm{d_j}}{2}\log \mathrm{n} \tag{8.45}$$

It is similar to the AIC but it has a stronger penalization and tends to choose simpler models.

Using priors $\pi_{\mathrm{j}}(\theta_{\mathrm{j}})$ on the parameters θ_{j} and a probability $\mathrm{p_j}$ indicating the probability that $\mathcal{M}_{\mathrm{j}}$ is the true model, by Bayesian theory, indicating with $\mathbf{Z}$ the training set, we have:

$$\mathrm{P}(\mathcal{M}_{\mathrm{j}}|\mathbf{Z}) \propto \mathrm{p}(\mathbf{Z}|\mathcal{M}_{\mathrm{j}})\mathrm{p_j} \tag{8.46}$$

Furthermore

$$\begin{aligned} \mathrm{p}(\mathbf{Z}|\mathcal{M}_{\mathrm{j}}) &= \int \mathrm{p}(\mathbf{Z}|\mathcal{M}_{\mathrm{j}}, \theta_{\mathrm{j}})\pi_{\mathrm{j}}(\theta_{\mathrm{j}})\mathrm{d}\theta_{\mathrm{j}} = \\ &= \int \mathrm{L}(\theta_{\mathrm{j}})\pi_{\mathrm{j}}(\theta_{\mathrm{j}})\mathrm{d}\theta_{\mathrm{j}} \end{aligned} \tag{8.47}$$

So equivalently we choose j to maximize:

$$\log \int \mathrm{L}(\theta_{\mathrm{j}})\pi_{\mathrm{j}}(\theta_{\mathrm{j}})\mathrm{d}\theta_{\mathrm{j}} + \log \mathrm{p_j} \tag{8.48}$$

Using Taylor expansion:

$$\begin{aligned} \log \int \mathrm{L}(\theta_{\mathrm{j}})\pi_{\mathrm{j}}(\theta_{\mathrm{j}})\mathrm{d}\theta_{\mathrm{j}} + \log \mathrm{p_j} &\approx \ell_{\mathrm{j}}(\hat{\theta}_{\mathrm{j}}) - \frac{\mathrm{d_j}}{2}\log \mathrm{n} \\ \Rightarrow \mathrm{BIC(j)} &= \ell_{\mathrm{j}}(\hat{\theta}_{\mathrm{j}}) - \frac{\mathrm{d_j}}{2}\log \mathrm{n} \end{aligned} \tag{8.49}$$

The prior has been ignored because it can be shown that the terms involving the prior are lower orders than the ones appearing in 8.49. Again in some books the signs are inverted.

The difference from AIC and Cross-Validation is that BIC assumes that one of the models is true and that we are trying to find the model most likely to be true in a Bayesian sense. AIC and Cross-Validation tries to find the model that predicts the best.

Under the Gaussian model, assuming the variance σ_ϵ^2 is known, with a squared error loss:

$$\begin{aligned} \bar{\mathrm{err}} &= \sum_{\mathrm{i}} \Big(\mathrm{y_i} - \hat{\mathrm{f}}(\mathrm{x_i})\Big)^2 \\ 2\ell_{\mathrm{j}}(\hat{\theta}_{\mathrm{j}}) &= -\mathrm{N}\frac{\bar{\mathrm{err}}}{\sigma_\epsilon^2} \\ \mathrm{BIC(j)} &= -\frac{\mathrm{N}}{2\sigma_\epsilon^2}\bar{\mathrm{err}} - \frac{\mathrm{d_j}}{2}\log \mathrm{n} = \frac{\mathrm{N}}{2\sigma_\epsilon^2}\left[\frac{\mathrm{d_j}}{\mathrm{N}}\sigma_\epsilon^2 \log \mathrm{N} - \bar{\mathrm{err}}\right] \end{aligned} \tag{8.50}$$

Hence BIC is proportional to AIC with the factor 2 replaced by $\log \mathrm{N}$.

8.6 Minimum Description Length

The minimum description length (MDL) principle is a formalization of Occam's razor in which the best hypothesis (a model and its parameters) for a given set of data is the one that leads to the best compression of the data. The Minimum Description Length is the minimum number of bits, or the minimum of the sum of the number of bits required to represent the data and the model. $\cdots$

8.7 Cross-Validation

It is the simplest and most widely used widely used method for estimating prediction error is cross-validation. This method directly estimates the expected extra-sample error. It consists of dividing data set in parts: a training set and using an independent test sample from the joint distribution. So one set is used for training and the other to avoid overfitting (to validate the model).

8.7.1 K-fold cross-validation

K-fold cross-validation uses part of the available data to fit the model, and a different part to test it. For the $\mathrm{k}-\mathrm{th}$ part, we fit the model using the other $\mathrm{K}-1$ parts for training and we calculate the prediction error of the fitted model on the $\mathrm{k}-\mathrm{th}$ part. We do this for all $\mathrm{k} = 1, \cdots, \mathrm{K}$, fitting the model resulting from the previous step using the new partition of data. Typical choices of K are 5 or 10. $\mathrm{K} = \mathrm{N}$ case is known as live-one-out

cross-validation. The higher value of K leads to less biased model (but large variance might lead to overfit), where as the lower value of K is similar to the train-test split approach we saw before.

8.8 Bootstrap Methods

The general idea is to randomly draw datasets with replacement from training data. Samples are constructed by drawing observations from a large data sample one at a time and returning them to the data sample after they have been chosen. This allows a given observation to be included in a given small sample more than once. This approach to sampling is called sampling with replacement. This is done B times, producing B datasets.

For the process we need to decide the number of bootstrap samples and a sample size. Then for each boostrap sample, we refit the model to each of the bootstrap datasets and examine the behaviour of the fits over the B replications.

[From wiki:] As an example, assume we are interested in the average (or mean) height of people worldwide. We cannot measure all the people in the global population, so instead we sample only a tiny part of it, and measure that. Assume the sample is of size N; that is, we measure the heights of N individuals. From that single sample, only one estimate of the mean can be obtained. In order to reason about the population, we need some sense of the variability of the mean that we have computed. The simplest bootstrap method involves taking the original data set of N heights, and, using a computer, sampling from it to form a new sample (called a 'resample' or bootstrap sample) that is also of size N. The bootstrap sample is taken from the original by using sampling with replacement (e.g. we might 'resample' 5 times from [1,2,3,4,5] and get [2,5,4,4,1]), so, assuming N is sufficiently large, for all practical purposes there is virtually zero probability that it will be identical to the original "real" sample. This process is repeated a large number of times (typically 1,000 or 10,000 times), and for each of these bootstrap samples we compute its mean (each of these are called bootstrap estimates). We now can create a histogram of bootstrap means. This histogram provides an estimate of the shape of the distribution of the sample mean from which we can answer questions about how much the mean varies across samples.

9 Additive models

A linear model has the form $\alpha + f_1(X_1) + f_2(X_2) + f_3(X_3) + \cdots$, where X are the predictors.

9.1 Tree based methods

Tree-based methods partition the feature space into a set of rectangles, and then fit a simple model (like a constant) in each one.

We restrict attention to recursive binary partitions. We first split the space into two regions, and model the response by the mean of Y in each region. We choose the variable and split-point to achieve the best fit. Then one or both of these regions are split into two more regions, and this process is continued, until some stopping rule is applied.

For example we first split at $X_1 = t_1$. Then the region $X_1 \leq t_1$ is split at $X_2 = t_2$ and the region $X_1 > t_1$ is split at $X_1 = t_3$ and so on. To each sub-region we assign a constan as output:

$$\hat{f}(X) = \sum_{m=1}^{M} c_m I\{(\mathbf{x} \in \mathcal{R}_m\} \tag{9.1}$$

An advantage of this representation is its interpretability. When more than 2 inputs are involved, it is difficult to represent the partition of the input space.

9.1.1 Regression Trees

The algorithm needs to decide automatically the splitting variables and points. We can start defining the values of the constant of each region:

$$\hat{c}_m = \text{ave}(y_i | x_i \in \text{CMcal}R_m) \tag{9.2}$$

Now finding the best binary partition in terms of minimum sum of squares is generally computationally infeasible. Hence we proceed with a greedy algorithm. Consider a splitting variable j and split point s and define the pair of half-planes:

$$R_1(j, s) = \{X | X_j \leq s\}. \quad R_2(j, s) = \{X | X_j > s\} \tag{9.3}$$

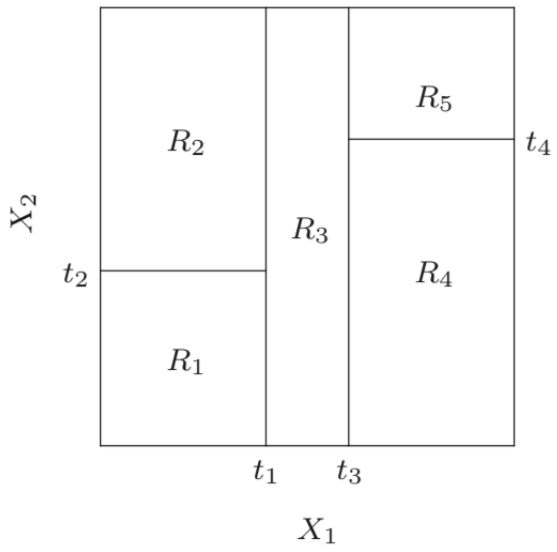


Figure 9.1: 2D-input space binary tree partition.

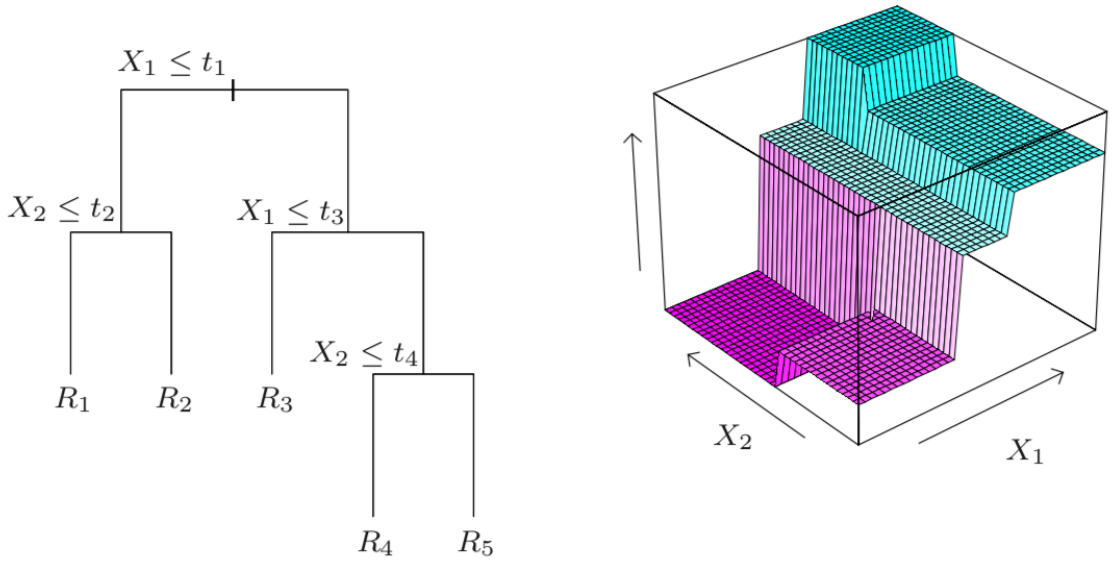


Figure 9.2: Another representation of the binary tree on the left. On the right the 3D resulting shape of the function.

Then we seek the splitting variable j and split point s that solve:

$$\begin{aligned}
&\min_{j,s}\left[\min_{c_1}\sum_{x_i\in\mathcal{R}_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in\mathcal{R}_2(j,s)}(y_i-c_2)^2\right] \\
&\hat{c}_1=\text{ave}(y_i|x_i\in R_1(j,s)) \\
&\hat{c}_2=\text{ave}(y_i|x_i\in R_2(j,s))
\end{aligned} \tag{9.4}$$

For each splitting variable, the determination of the split point s can be done very quickly and hence by scanning through all of the inputs, determination of the best pair (j,s) is feasible. Having found the best split, we partition the data into the two resulting regions and repeat the splitting process on each of the two regions. Then this process is repeated on all of the resulting regions.

Tree size is a tuning parameter governing the model's complexity, and the optimal tree size should be adaptively chosen from the data. One approach would be to split tree nodes only if the decrease in sum-of-squares due to the split exceeds some threshold. This strategy is too short-sighted, however, since a seemingly worthless split might lead to a very good split below it. The preferred strategy is to grow a large tree T_0, stopping the splitting process only when some minimum node size (say 5) is reached. Then this large tree is pruned using **cost-complexity pruning**.

The **cost-complexity pruning** is the following. We define a subtree T of T_0. obtained by collapsing any number of its internal nodes. We index the terminal nodes by m. let $|T|$ denote the number of terminal nodes in T and

$$\begin{aligned}
N_m&=\#\{x_i\in\mathcal{R}_m\} \\
\hat{c}_m&=\frac{1}{N_m}\sum_{x_i\ \text{in}\mathcal{R}_m}y_i \\
Q_m(T)&=\frac{1}{N_m}\sum_{x_i\in R_m}(y_i-\hat{c}_m)^2
\end{aligned} \tag{9.5}$$

We define the cost complexity criterion:

$$C_\alpha(T)=\sum_{m=1}^{|T|}N_mQ_m(T)+\alpha|T| \tag{9.6}$$

The idea is to find for each α the subtree T_α of T_0 to minimize such a function. For each α there is a unique smallest subtree T_α that minimizes $C_\alpha(T)$. To find T_α we use the **weakest link pruning**: we collapse the internal node that produces the smallest per-node increase and we continue umtil we produce the single-node (root) tree. So we have a sequence of subtrees that must contain T_α. We choose α to minimize the cross-validated sum of squares, obtaining the tree T_α.

9.2 Classification trees

The procedure is very similar. In a node m representing the region R_m with N_m observations, let

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k) \tag{9.7}$$

We classify the observations in node m to class

$$k(m) = \text{argmax}_k \hat{p}_{mk}, \tag{9.8}$$

the majority class in node m. Different measures $Q_m(T)$ of node impurity exist

$$\begin{aligned}
&\textbf{Misclassification error} && \frac{1}{N_m} \sum_{i \in CMcalR_m} I(y_i \neq k(m)) = 1 - \hat{p}_{km} \\
&\textbf{Gini index} && \sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \\
&\textbf{Cross-entropy or deviance} && -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}
\end{aligned} \tag{9.9}$$

These trees are referred as CART (classification and regression trees).

9.2.1 Linear Combination splits

Rather than using splits of the type $X_j \leq s$ we can use splits of the type $\sum a_j X_j \leq s$. The weights a_j and split point s are optimized to minimize the relevant criterion such as the Gini index.

9.2.2 Limitations of trees

One major problem of trees is **the instability (i.e., high variance)**. A small change in the data can result in a very different series of splits, making interpretation somewhat precarious. The major reason is the hierarchical nature of the process: the effect of an error in the top split is propagated down to all the next splits. The problem can be attenuated but not completely solved.

Another problem of trees is the **lack of smoothness**. This is not a problem in classification but it is in regression.

Also they are not good at capturing additive structures, such as $Y = c_1(X_1 < t_1) + c_1(X_1 < t_1) + \epsilon$ where the noise is 0 mean. Then a binary tree might make its first split on X_1 near t_1. At the next level down it would have to split both nodes X_2 at t_2 in order to capture the additive structure. This is achieved with sufficient data, but it it is a fragile mechanism and can fail in case of many additive effects.

9.3 Patient rule induction method (PRIM)

Also PRIM finds boxes in the feature space, but it seeks boxes in which the response average is high. Hence it looks for maxima in the target function, an exercise known as bump hunting. The main box construction method in PRIM works starting with a box containing all of the data. The box is compressed along one face by a small amount, and peeling off a constant portion of data denoted by α. Typically α is 0.05 and 0.1 of the remaining points at each stage. The face chosen for compression is the one resulting in the largest box mean, after the compression is performed. Then the process is repeated, stopping when the current box contains some minimum number of data points.

After the top-down sequence is computed, PRIM reverses the process, expanding along any edge, if such an expansion increases the box mean. This is called pasting. The result of these steps is a sequence of boxes, with different numbers of observation in each box. Cross-validation, combined with the judgement of the data analyst, is used to choose the optimal box size. Denote by B_1 the points in this first box. These points are now removed from the training set and we repeat the two processes (top-down peeling and bottom-up pasting) several times, obtaining for each iteration a new box B_i.

Each box is defined by a set of rules involving a subset of predictors such as:

$$(a_1 \leq X_1 \leq b_1) \qquad (b_1 \leq X_3 \leq b_2) \tag{9.10}$$

PRIM can handle a categorical predictor by considering all partitions of the predictor, as in CART. Missing values are also handled in a manner similar to CART. PRIM is designed for regression

Start with all of the training data, and a maximal box containing all of the data;
Consider shrinking the box by compressing one face, so as to peel off the proportion α of observations having either the highest values of a predictor X_j or the lowest. Choose the peeling that produces the highest response mean in the remaining box;
Repeat step 2 until some minimal number of observations (say 10) remain in the box;
Expand the box along any face, as long as the resulting box mean increases;
Steps 1-4 give a sequence of boxes, with different numbers of observations in each box. Use cross-validation to choose a member of the sequence. Call the box B_1.
Remove the data in the box B_1 from the dataset and repeat steps 2-5 to obtain a second box. and continue to get as many boxes as desired.

PRIM is designed for binary regression. There is no simple way to deal with $k > 2$ classes simultaneously: it is possible to run PRIM separately for each class versus a baseline class.In any case, the ability of PRIM to be more patient should help the top-down greedy algorithm find a better solution.

9.4 Multivariate Adaptive Regression Splines

It is a procedure well-suited for high-dimensional problems. MARS uses expansions in piecewise linear basis functions of the form $(x-t)_+$ and $(t-x)_+$.

Each function is piecewise linear, with a knot at the value t_1. We call the two functions a **reflected pair**. The idea is to form reflected pairs for each input X_j with knots at each observed value x_{ij} of that input. If all of the input values are distinct, there are 2Np basis functions altogether. The model-building strategy is like a forward stepwise linear regression, but

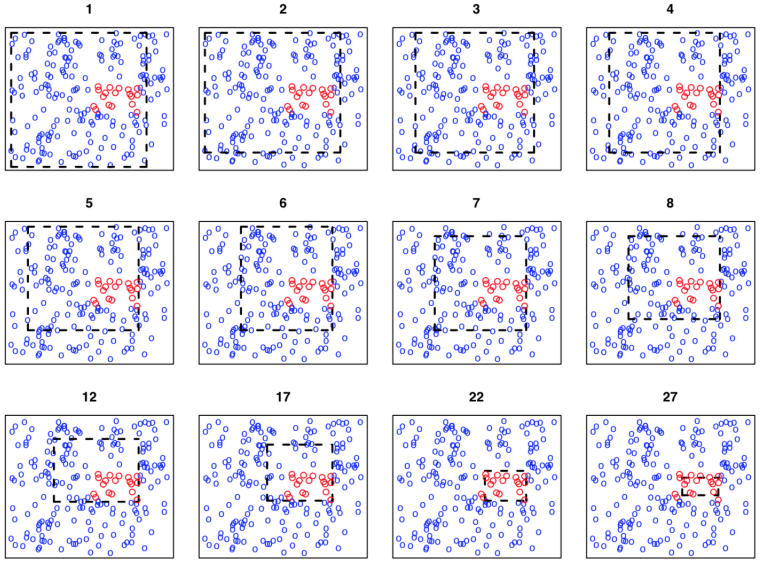


Figure 9.3: Steps in PRIM.

instead of using the original inputs, we are allowed to use functions from the set $\mathcal{C}$ and their products:

$$\mathcal{C} = \{(X_j - t)_+, (t - X_j)_+\}_{t \in \{x_{1j}, \cdots, x_{Nj}\}, j=1,\cdots,p} \tag{9.11}$$

Thus the model has the form:

$$f(X) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(X) \tag{9.12}$$

where each $h_m(X)$ is a function in $\mathcal{C}$ or a product of two or more such functions. Given the choice for $h_m(X)$, the coefficients are estimated with the residual sum of squares. We start with only a constant $h_0(X) = 1$ and all the functions are candidate functions. At each step we consider as a new basis function pair all products of a function h_m in the model set $\mathcal{M}$ with one of the reflected pairs in $\mathcal{C}$. We add to the model $\mathcal{M}$ the term of the form:

$$\hat{\beta}_{M+1} h_\ell(X)(X_j - t)_+ + \hat{\beta}_{M+2} h_\ell(X)(t - X_j)_+, h_\ell \in \mathcal{M} \tag{9.13}$$

that produces the largest decrease in training error. For example, at the first stage we consider adding to the model a function of the form: $\hat{\beta}_1(X_2 - x_{72})_+ +$

$\hat{\beta}_2(x_{72} - X_2)_+$. Then this pair of basis functions is added to the set $\mathcal{M}$ and at the next stage we consider including a pair of products in the form:

$$h_m(X)(X_j - t)_+ \qquad \text{and} \qquad h_m(X)(t - X_j)_+, t \in \{xij\} \tag{9.14}$$

where for h_m we have the choices:

$$\begin{aligned} &h_0(X) = 1 \\ &h_1(X) = (X_2 - x_{72})_+, \quad or \\ &h_2(X) = (x_{72} - X_2)_+ \end{aligned} \tag{9.15}$$

At the end of this process we have a large model. This model typically overfits the data, and so a backward deletion procedure is applied. The term whose removal causes the smallest increase in residual squared error is deleted from the model at each stage, producing an estimated best model $\hat{f}_\lambda$ of each size (number of terms) λ. One could use cross-validation to estimate the optimal value of λ, but for computational savings the MARS procedure instead uses generalized cross-validation:

$$GCV(\lambda) = \frac{\sum_{i=1}^{N}(y_i - \hat{f}_\lambda(x_i))^2}{(1 - \frac{M(\lambda)}{N})^2} \tag{9.16}$$