Machine Learning: Linear Models

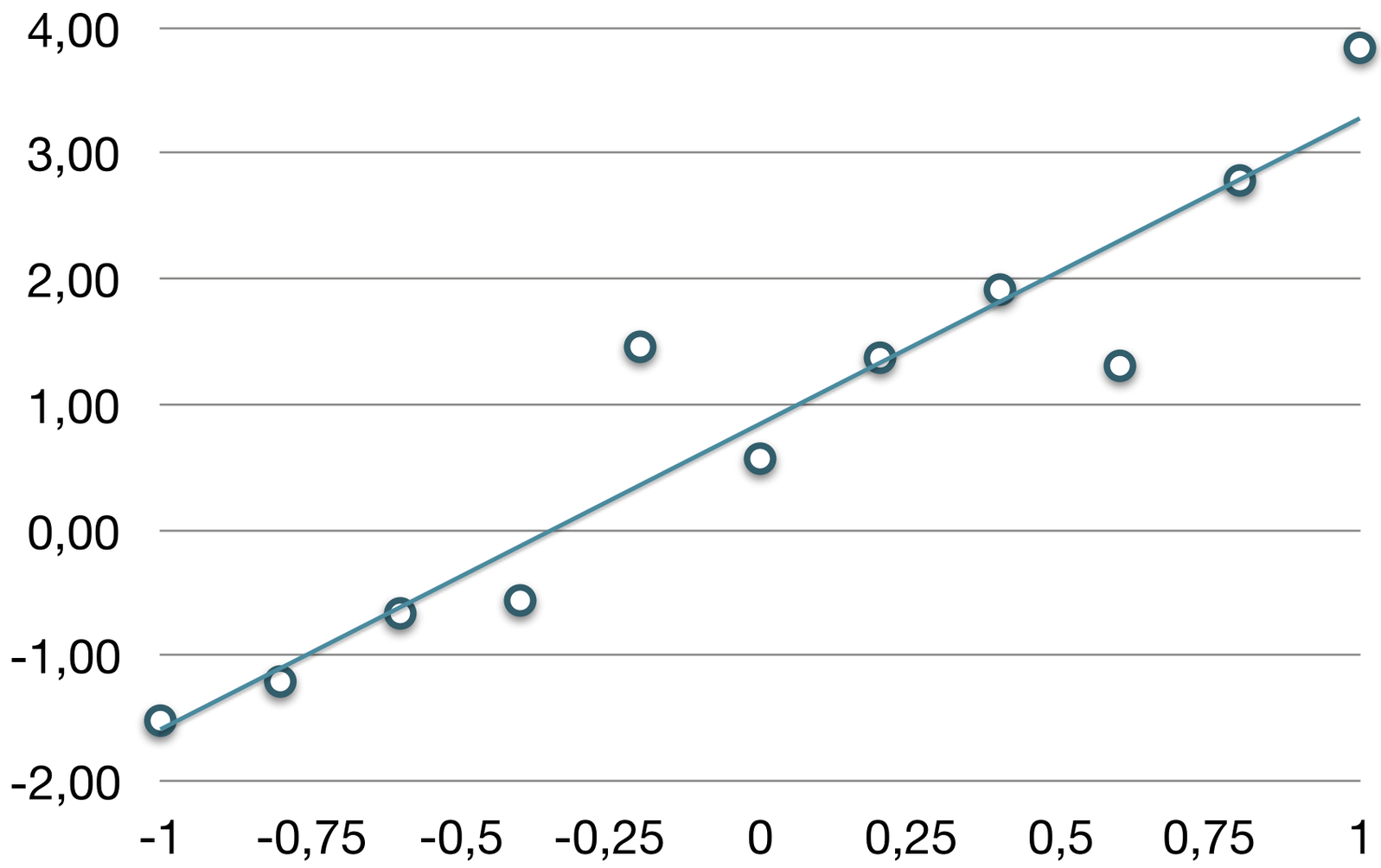
Roberto Esposito
roberto.esposito@unito.it

Least squares

The treatment of this topic is significantly different from the one taken in the book. I find the approach proposed here simpler and clearer. The book treatment however has its merits offering insights that are not apparent here. I suggest the interested students to have a look at the book contents even though that material will not be part of the mandatory readings.

Part of this material is based on Gilbert Strang linear algebra lessons (freely available on iTunesU and at https://ocw.mit.edu/courses/mathematics/18-06-linear-algebra-spring-2010/video-lectures/).

Best fitting line



Matrix representation

x	y
-1	-1,52
-0,8	-1,21
-0,6	-0,67
-0,4	-0,56
-0,2	1,46
0	0,57
0,2	1,37
0,4	1,91
0,6	1,31
0,8	2,78
1	3,84

We want to find the parameters *(C,D)* of the linear model $Cx + D = y$ that best fits the data.

Ideally we would like that, for each row of the table on the left, the linear equation above would hold. I.e. we aim at finding *(C,D)* such that:

$$\begin{cases} -1C + D = -1.52 \\ -0.8C + D = -1.21 \\ -0.6C + D = -0.67 \\ \dots \end{cases}$$

Matrix equation

So we end up with the system:

$$\begin{cases} -1C + D = -1.52 \\ -0.8C + D = -1.21 \\ -0.6C + D = -0.67 \\ \dots \end{cases}$$

corresponding to the matrix equation:

$$\underset{\mathbf{X}}{\begin{pmatrix} -1 & 1 \\ -0.8 & 1 \\ -0.6 & 1 \\ \dots & \dots \end{pmatrix}} \cdot \underset{\mathbf{w}}{\begin{pmatrix} C \\ D \end{pmatrix}} = \underset{\mathbf{y}}{\begin{pmatrix} -1.52 \\ -1.21 \\ -0.67 \\ \dots \end{pmatrix}}$$

i.e.,: $\mathbf{Xw} = \mathbf{y}$

Solving the matrix equation — ideal case

In the ideal case $\mathbf{X}$ is squared and full rank, and the equation can be solved simply by multiplying both sides by the inverse of $\mathbf{X}$.

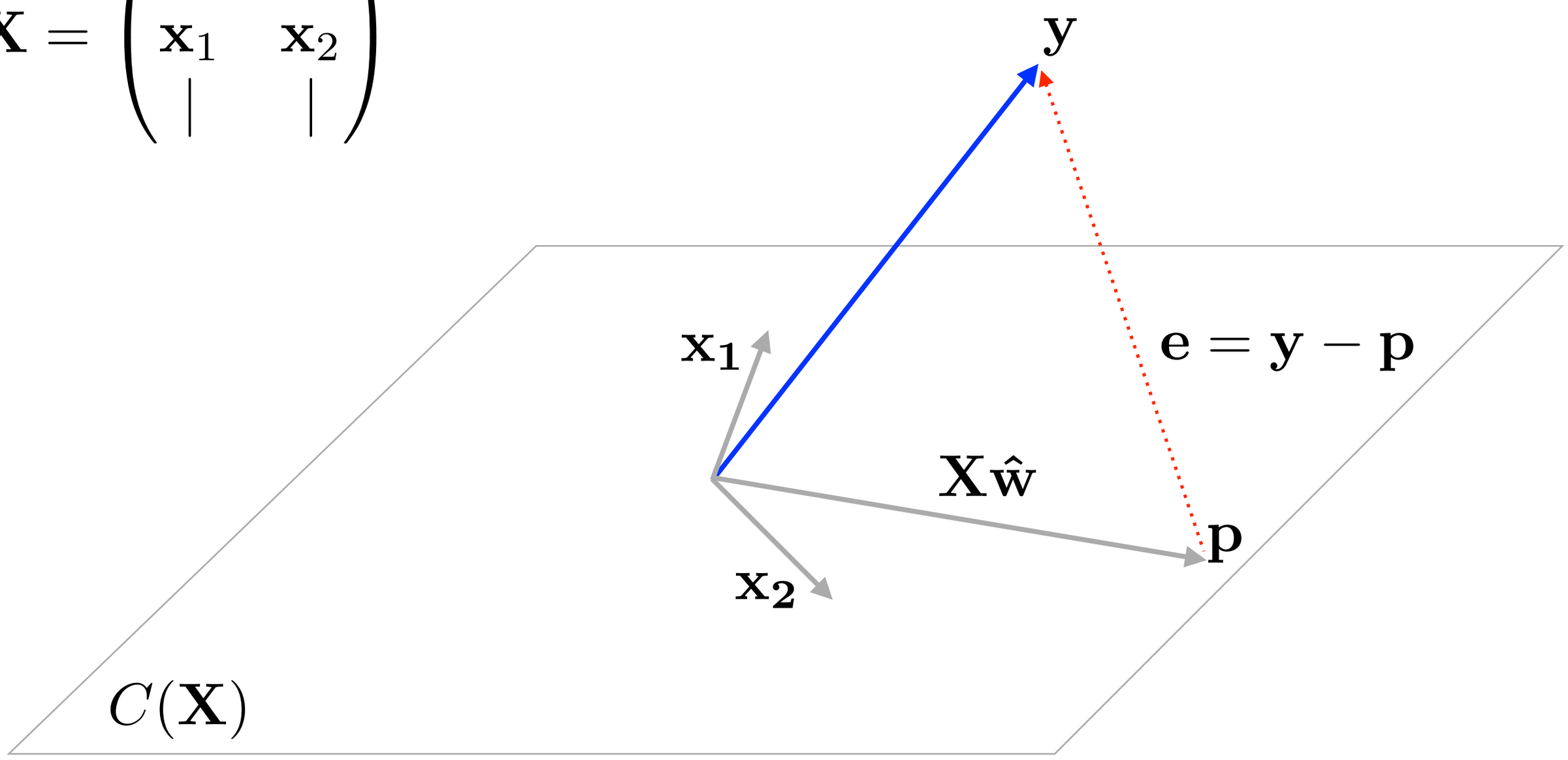
$$\mathbf{w} = \mathbf{X}^{-1} \cdot \mathbf{y}$$

Unfortunately, this is true only in the most trivial cases. $\mathbf{X}$ is not usually invertible and a more general approach is needed.

We start by noticing that by varying $\mathbf{w}$ we are actually spanning the column space of $\mathbf{X}$ and that $\mathbf{y}$ does not belong to it (otherwise we could solve for equality).

Relationship between $\mathbf{x}$ and estimation errors

$$\mathbf{X} = \begin{pmatrix} | & | \\ \mathbf{x}_1 & \mathbf{x}_2 \\ | & | \end{pmatrix}$$



Least squares as a minimisation problem

Error: $\|\mathbf{e}\|_2 = \|\mathbf{y} - \mathbf{p}\|_2 = \sqrt{\sum_i (y_i - p_i)^2}$

We can then formulate the problem as the one of minimising the norm of $\mathbf{e}$, *which is equivalent to* ***minimise*** *the sum of the* ***squared differences*** *between* components of $\mathbf{y}$ and $\mathbf{p}$.

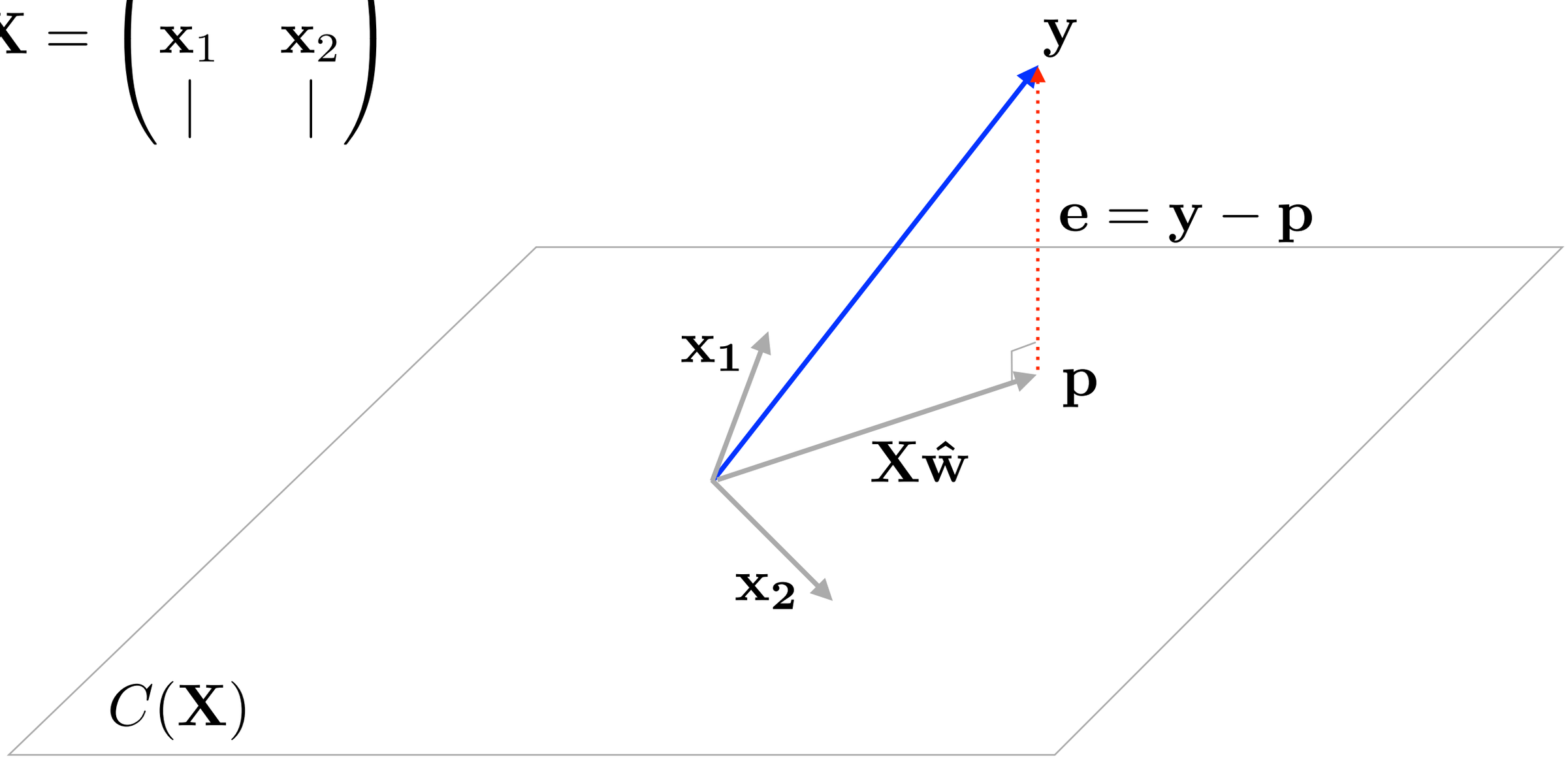
Since, by definition: $\mathbf{p} = \mathbf{X}\hat{\mathbf{w}}$

It follows that the whole problem can be formulated as

$$\text{minimize}_{\hat{\mathbf{w}}} \|\mathbf{X}\hat{\mathbf{w}} - \mathbf{y}\|_2^2$$

Relationship between $\mathbf{x}$ and estimation errors

$$\mathbf{X} = \begin{pmatrix} | & | \\ \mathbf{x}_1 & \mathbf{x}_2 \\ | & | \end{pmatrix}$$



Least squares solution

Imposing orthogonality of $\mathbf{e}$ and $\mathbf{C(X)}$ is equivalent to impose:

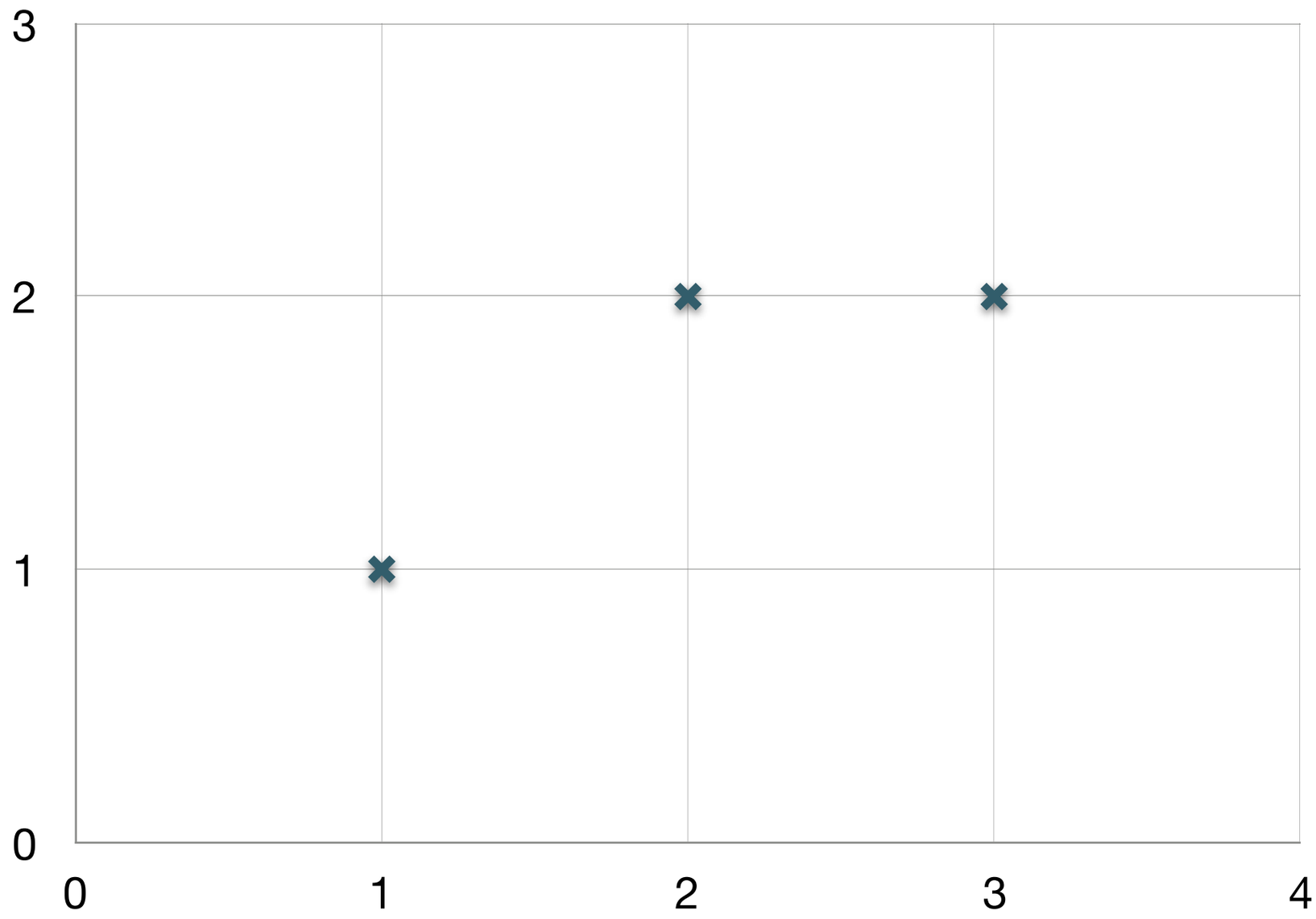
$$\mathbf{X}^T\mathbf{e} = 0$$

implying:

$$\begin{aligned}
& \mathbf{X}^T(\mathbf{y} - \mathbf{X}\hat{\mathbf{w}}) = 0 \\
\Rightarrow \quad & \mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{X}\hat{\mathbf{w}} = 0 \\
\Rightarrow \quad & \mathbf{X}^T\mathbf{X}\hat{\mathbf{w}} = \mathbf{X}^T\mathbf{y} \\
\Rightarrow \quad & \hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}
\end{aligned}$$

Example using three points

x	y
1	1
2	2
3	2



Example: Matrix formulation

x	y
1	1
2	2
3	2

$$\mathbf{X} = \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix} \qquad \mathbf{y} = \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix}$$

Example: calculating solution

Example: calculating solution

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Example: calculating solution

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

$$\mathbf{X}^T\mathbf{X} = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix} = \begin{pmatrix} 14 & 6 \\ 6 & 3 \end{pmatrix}$$

Example: calculating solution

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

$$\mathbf{X}^T\mathbf{X} =$$

$$\left(\begin{array}{cc|cc} 14 & 6 & 1 & 0 \\ 6 & 3 & 0 & 1 \end{array}\right) \rightarrow \left(\begin{array}{cc|cc} 2 & 0 & 1 & -2 \\ 6 & 3 & 0 & 1 \end{array}\right)$$

$$\rightarrow \left(\begin{array}{cc|cc} 1 & 0 & \frac{1}{2} & -1 \\ 6 & 3 & 0 & 1 \end{array}\right) \rightarrow \left(\begin{array}{cc|cc} 1 & 0 & \frac{1}{2} & -1 \\ 0 & 3 & -3 & 7 \end{array}\right)$$

$$\rightarrow \left(\begin{array}{cc|cc} 1 & 0 & \frac{1}{2} & -1 \\ 0 & 1 & -1 & \frac{7}{3} \end{array}\right)$$

$$(\mathbf{X}^T\mathbf{X})^{-1}$$

Example: calculating solution

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

$$\mathbf{X}^T\mathbf{X} = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix} = \begin{pmatrix} 14 & 6 \\ 6 & 3 \end{pmatrix}$$

$$(\mathbf{X}^T\mathbf{X})^{-1} = \begin{pmatrix} \frac{1}{2} & -1 \\ -1 & \frac{7}{3} \end{pmatrix}$$

$$\mathbf{X}^T\mathbf{y} = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix} = \begin{pmatrix} 11 \\ 5 \end{pmatrix}$$

Example: calculating solution

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

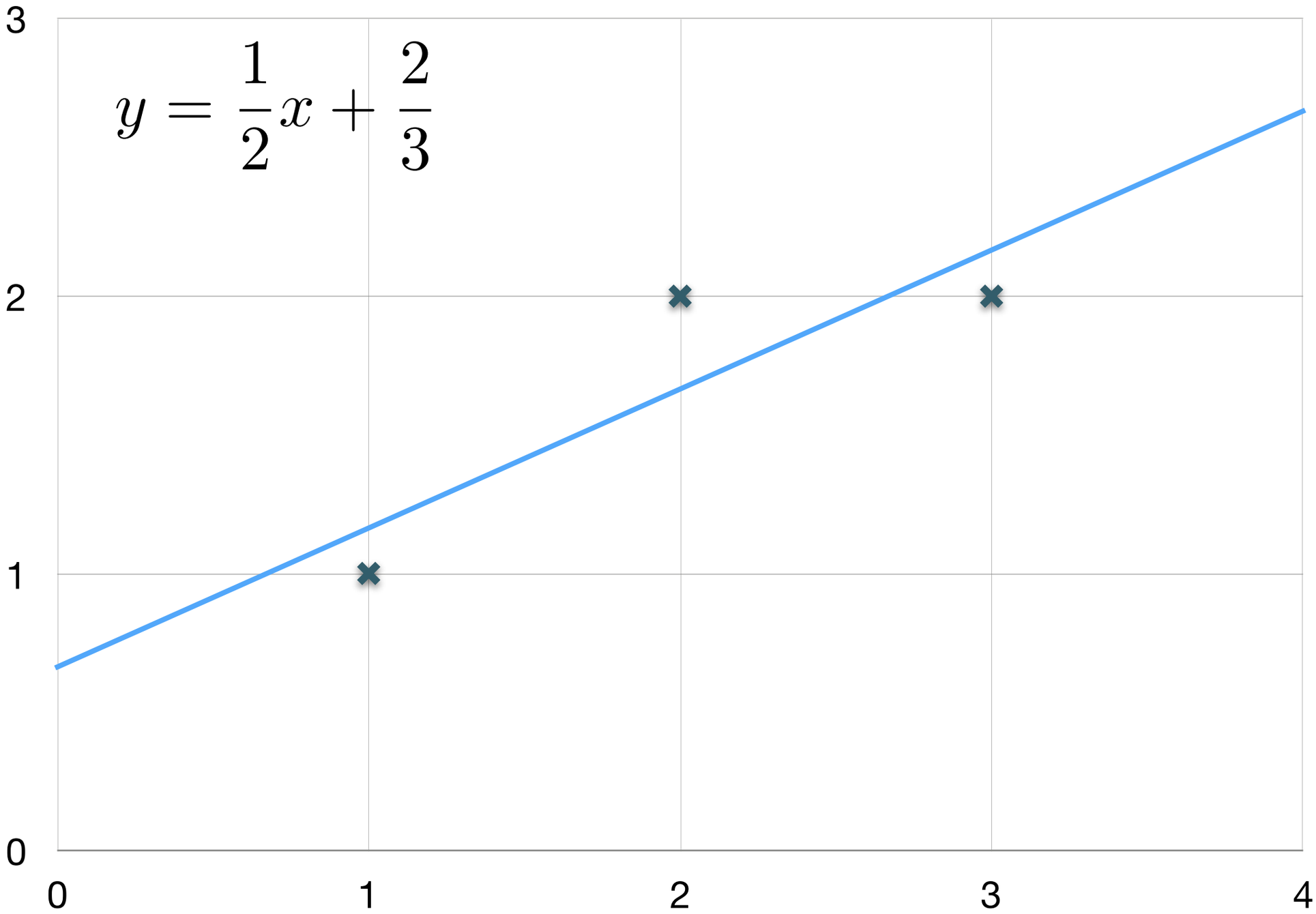
$$\mathbf{X}^T\mathbf{X} = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix} = \begin{pmatrix} 14 & 6 \\ 6 & 3 \end{pmatrix}$$

$$(\mathbf{X}^T\mathbf{X})^{-1} = \begin{pmatrix} \frac{1}{2} & -1 \\ -1 & \frac{7}{3} \end{pmatrix}$$

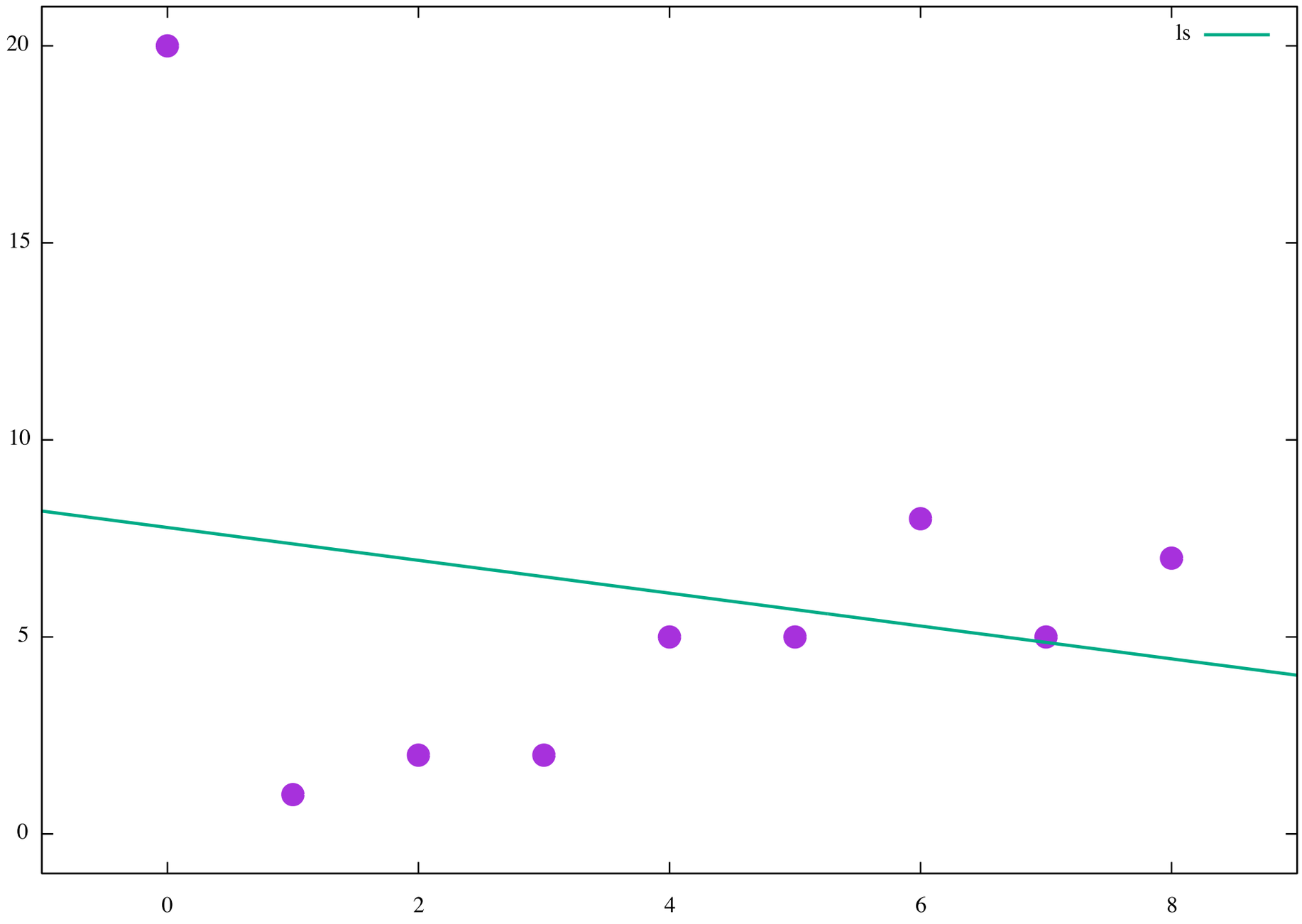
$$\mathbf{X}^T\mathbf{y} = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix} = \begin{pmatrix} 11 \\ 5 \end{pmatrix}$$

$$(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \begin{pmatrix} 1/2 & -1 \\ -1 & 7/3 \end{pmatrix} \begin{pmatrix} 11 \\ 5 \end{pmatrix} = \begin{pmatrix} 1/2 \\ 2/3 \end{pmatrix}$$

Example: least squares solution plot



Least Squares Overfitting



Regularised regression

Regularisation is a general method to avoid overfitting by applying additional constraints to the weight vector. A common approach is to make sure the weights are, on average, small in magnitude: this is referred to as **shrinkage**.

The regularised version of the least squares optimisation is:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} (\mathbf{y} - \mathbf{X}\mathbf{w})^{\mathrm{T}} (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda ||\mathbf{w}||^2$$

where λ is a scalar determining the amount of regularization.

Regularised regression

This regularised problem still has a closed-form solution:

$$\hat{\mathbf{w}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$$

where $\mathbf{I}$ denotes the identity matrix. Regularisation amounts to adding λ to the diagonal of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$, a well-known trick to improve the numerical stability of matrix inversion. This form of least-squares regression is known as **ridge regression.**

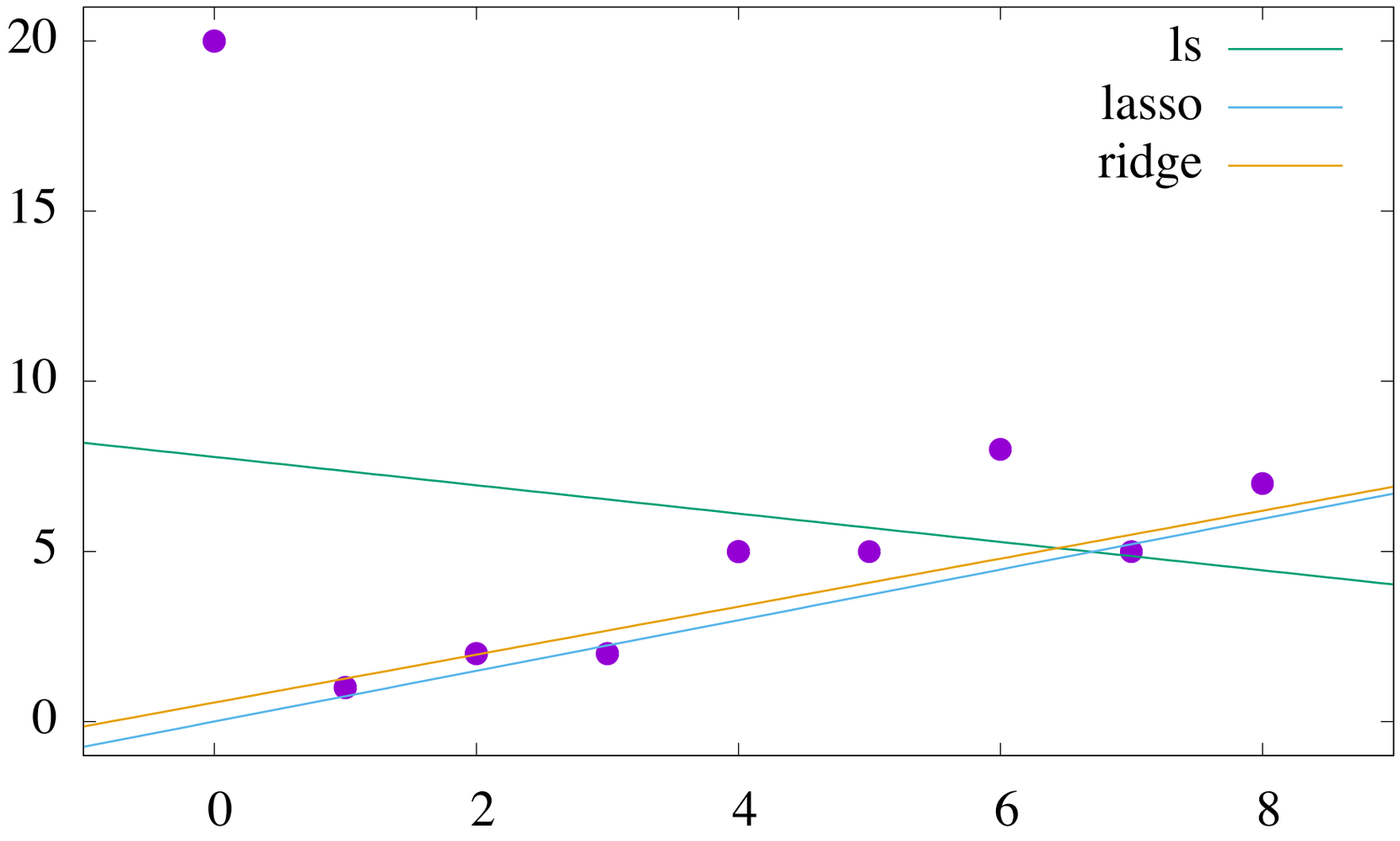
Regularised regression

An interesting alternative form of regularised regression is provided by the **lasso**, which stands for 'least absolute shrinkage and selection operator'. It replaces the ridge regularisation term $||\mathbf{w}||_2$ with the sum of absolute weights:

$$\|\mathbf{w}\|_1 = \sum_i |w_i|$$

The result is that some weights are shrunk, but others are set to 0, and so the lasso regression favours *sparse solutions*.

Example



$$\begin{matrix} \mathbf{w}_{\text{ls}} & \mathbf{w}_{\text{lasso}} & \mathbf{w}_{\text{ridge}} \end{matrix}$$
$$\begin{pmatrix} -0.41663 & 7.4465e-01 & 0.70500 \\ 7.77770 & 3.8439e-06 & 0.55758 \end{pmatrix}$$

Understanding regularisation

- Why does minimising the norm of the weight vector improve results?

Understanding regularisation

- Why does minimising the norm of the weight vector improve results?
- Why does the lasso approach favour sparse solutions?

Minimisation of the norm of the weight vector

Several reasons justify minimising the norm of the weight vector:

- If you assume that $\mathbf{X}$ is affected by an error $\mathbf{D}$ then:

$$(\mathbf{X} + \mathbf{D})\mathbf{w} = \mathbf{X}\mathbf{w} + \mathbf{D}\mathbf{w}$$

  and minimising the norm of the weight vector minimizes the effect of the error on $\mathbf{X}$.

- Smaller weight vectors can be thought as “simpler” models. In this sense minimising the norm of the weight vector is tantamount to following the Occam’s razor principle.

Minimisation of the norm of the weight vector (2)

- by imposing these additional constraints we are imposing a bias to the learning algorithm and this reduces the error variance: as a consequence of its quadratic loss, least squares is quite an unstable regressor: small variations in the data are emphasised by the loss.

By regularising it we are improving the error by reducing variance without hurting too much the bias component (surely it can be the case that the true function has a large $||\mathbf{w}||$ and that this is not reflected in the data, but we can assume this is an uncommon situation).

Norm minimisation and sparsity

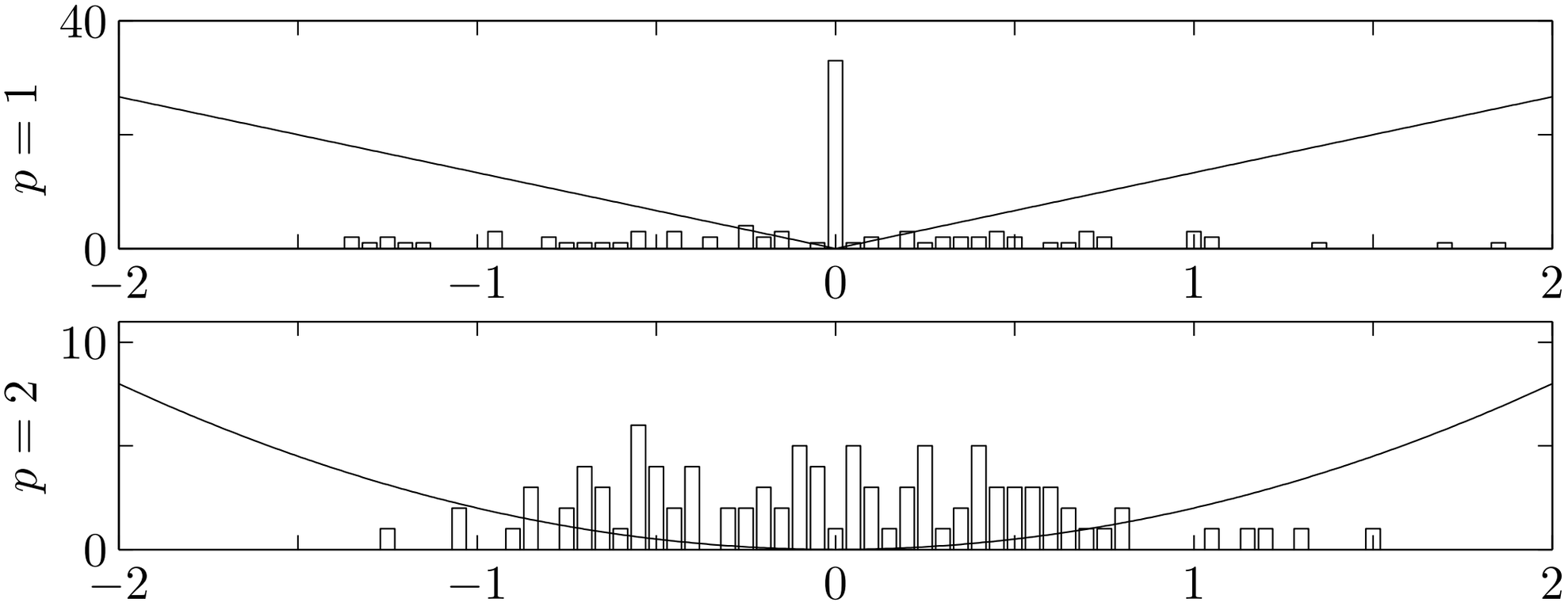


Image courtesy of **Stephen Boyd** and **Lieven Vandenberghe**, from the book "*Convex Optimization*". Cambridge University Press.

Least squares for classification

Using least squares for classification

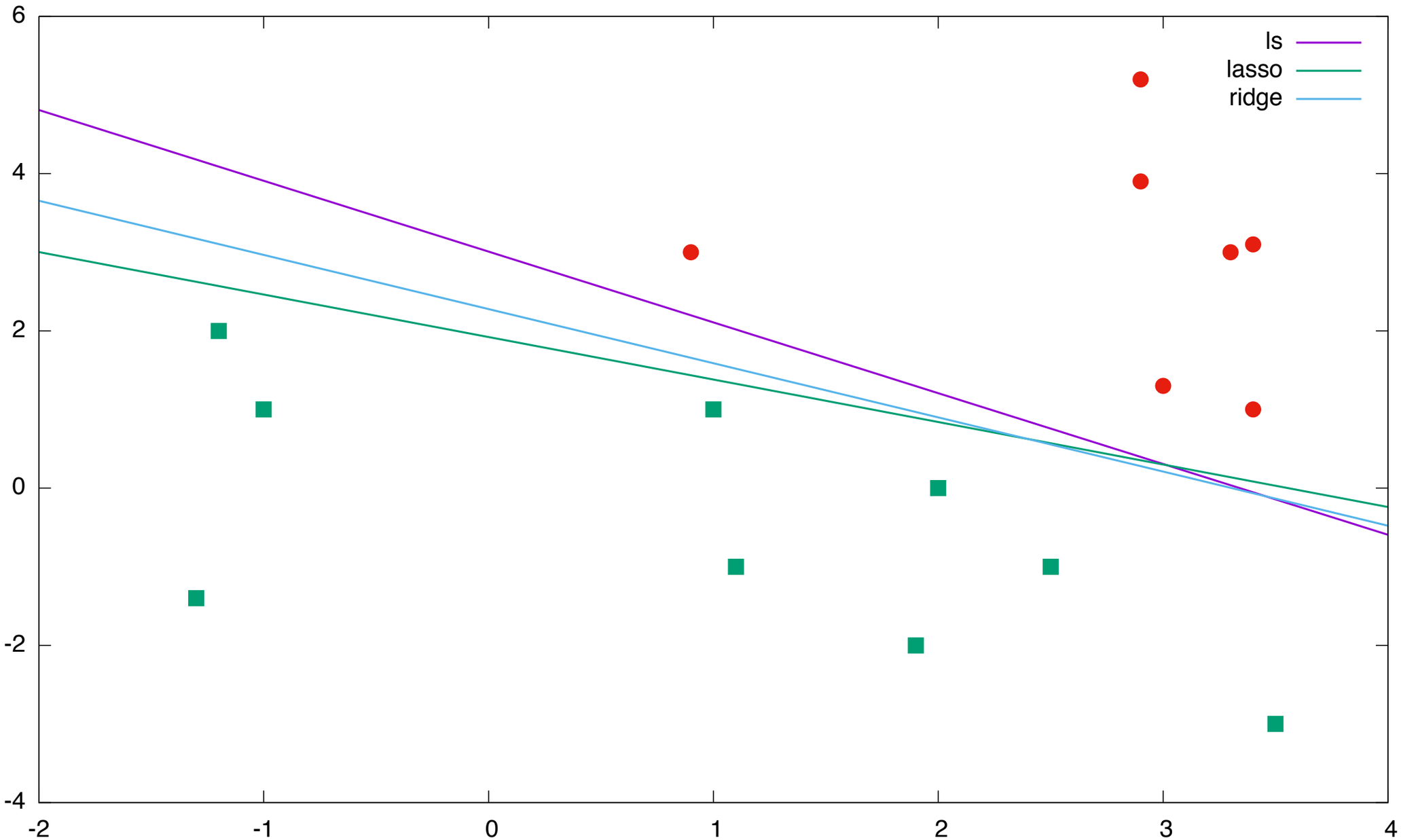
The least squares approach can be easily adapted to cope with classification tasks by representing the positive class using “1” and the negative class using “-1”.

Then:

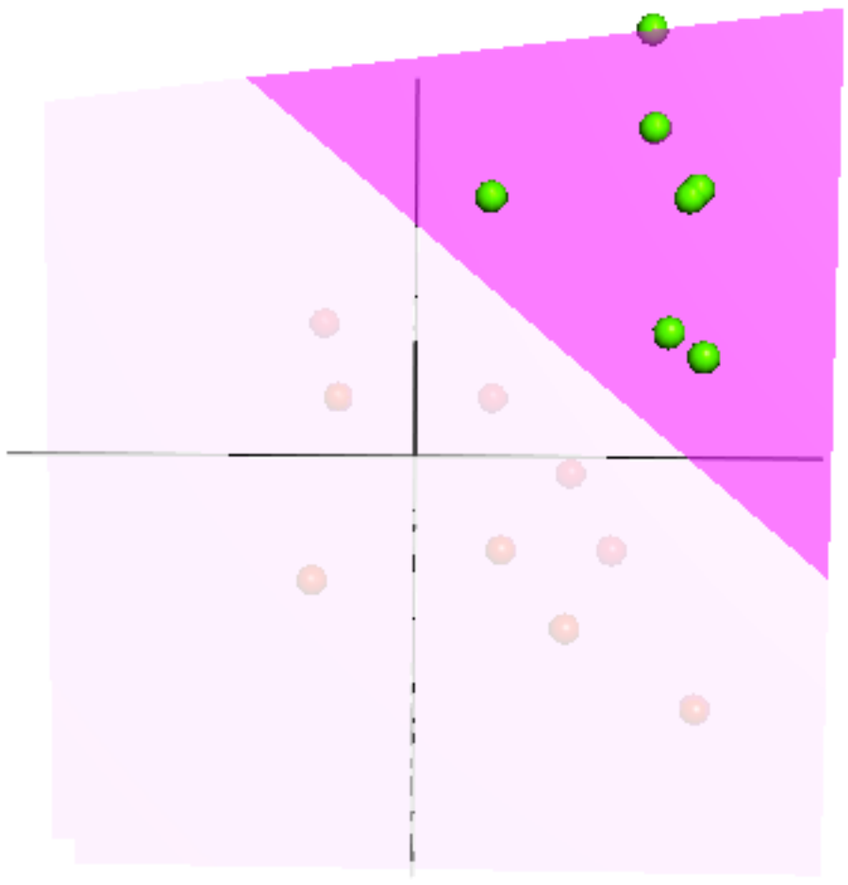
$$\hat{c}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x}^T\hat{\mathbf{w}} - t > 0 \\ 0 & \text{if } \mathbf{x}^T\hat{\mathbf{w}} - t = 0 \\ -1 & \text{if } \mathbf{x}^T\hat{\mathbf{w}} - t < 0 \end{cases}$$

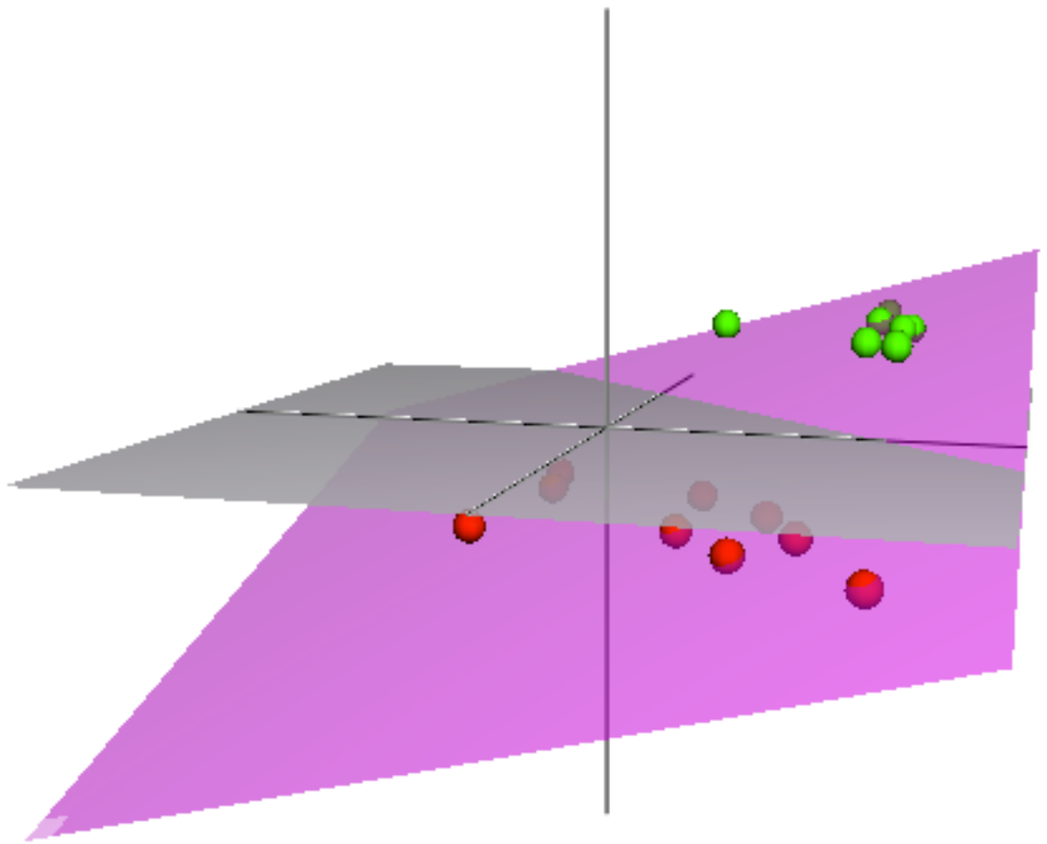
Note: t represents the intercepts. In previous slides this was included in the parameter vector and $\mathbf{x}$ was assumed to contain a 1 in the last position.

Example



Example

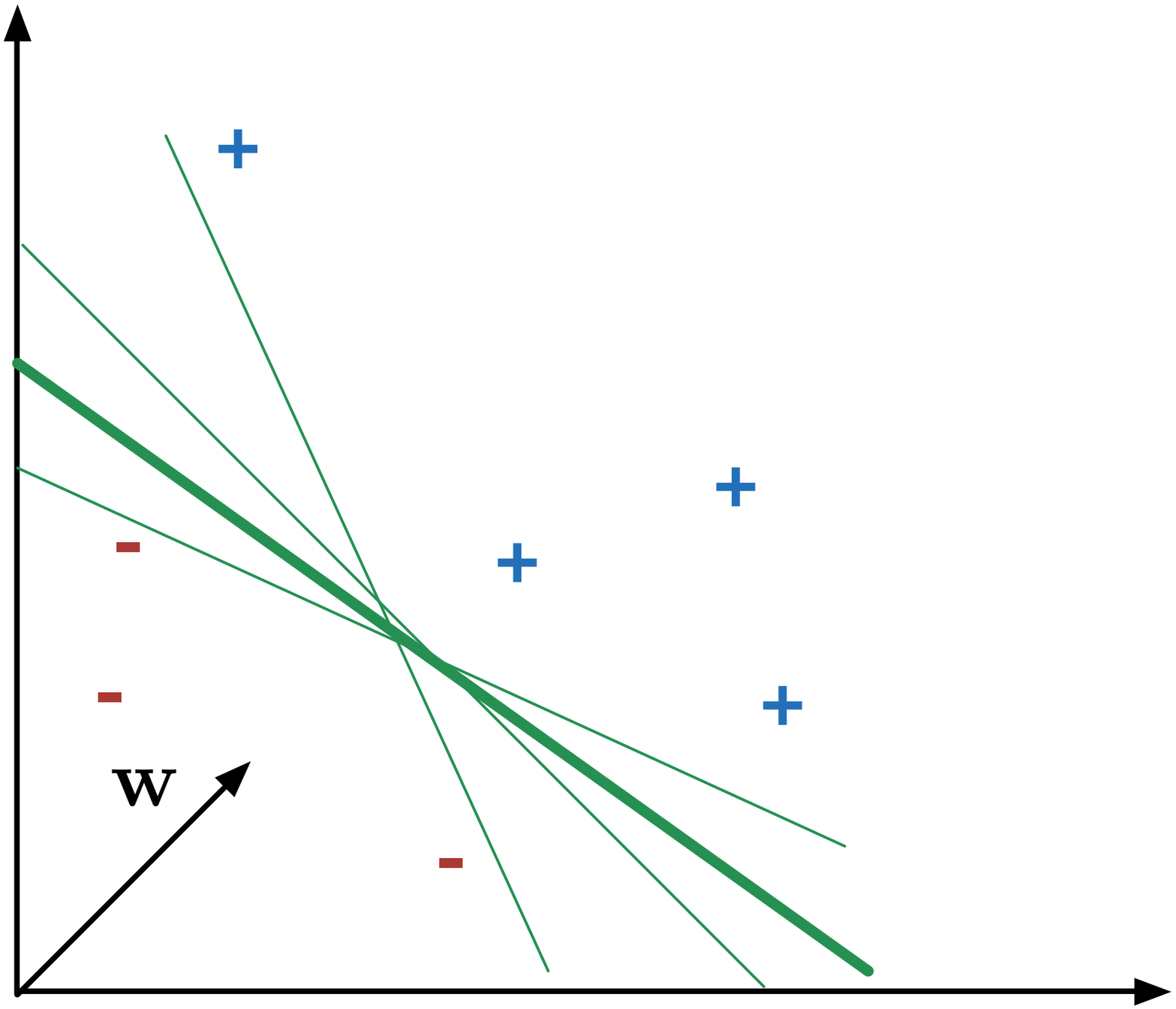




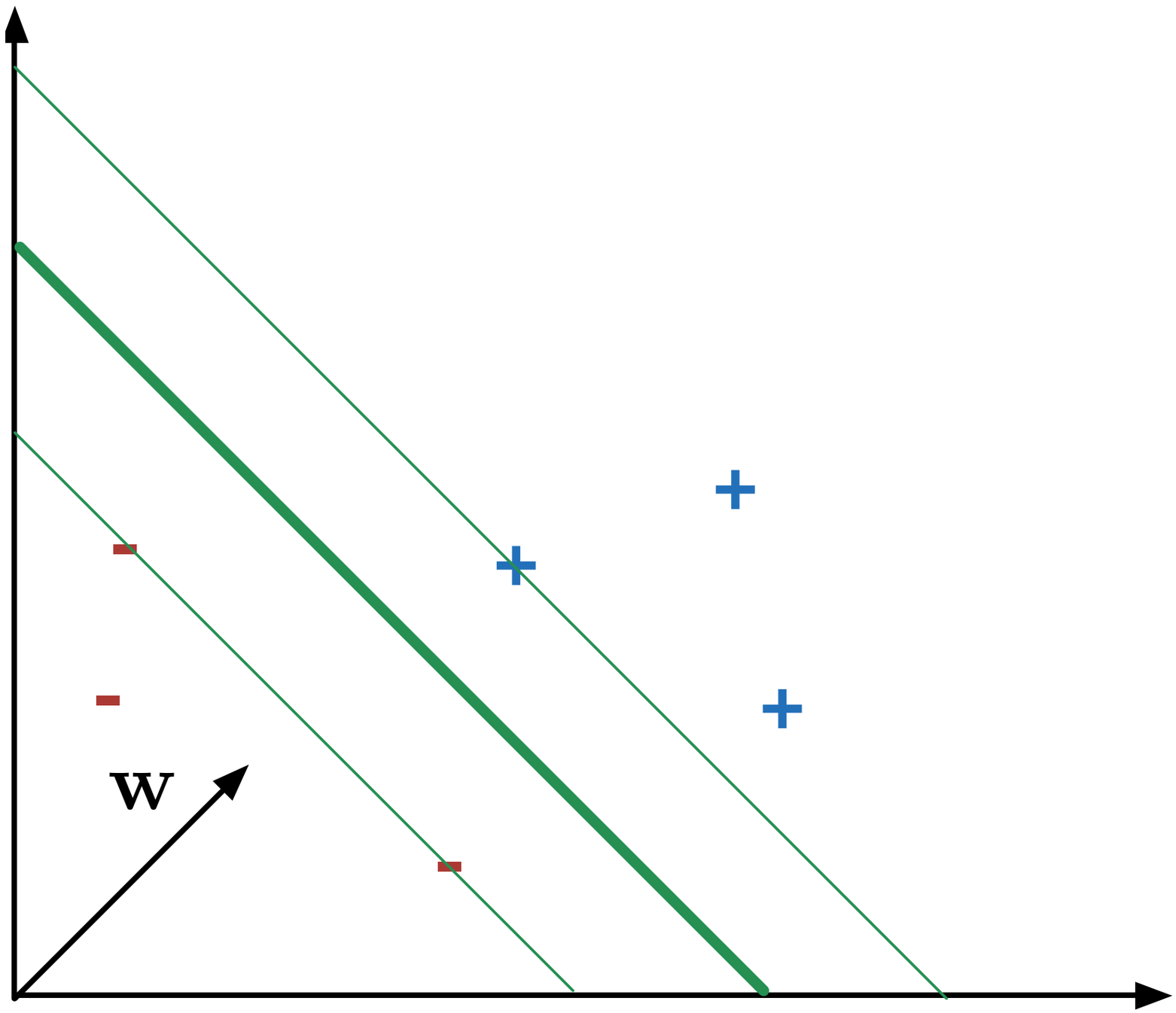
Support Vector Machines

The treatment of this topic is *in part* different from the one taken in the book.

SVM: finding the hyperplane maximising the margin

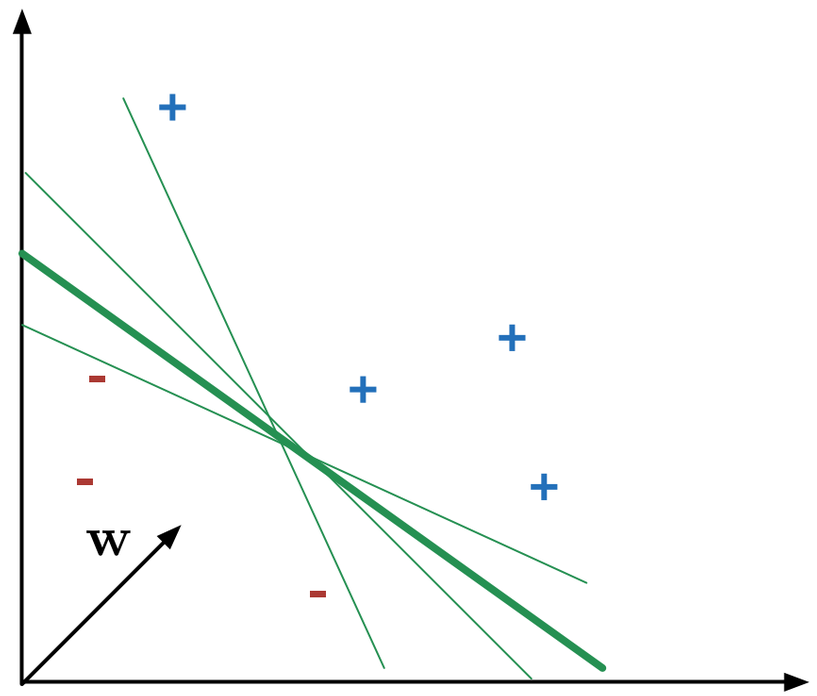


SVM: finding the hyperplane maximising the margin

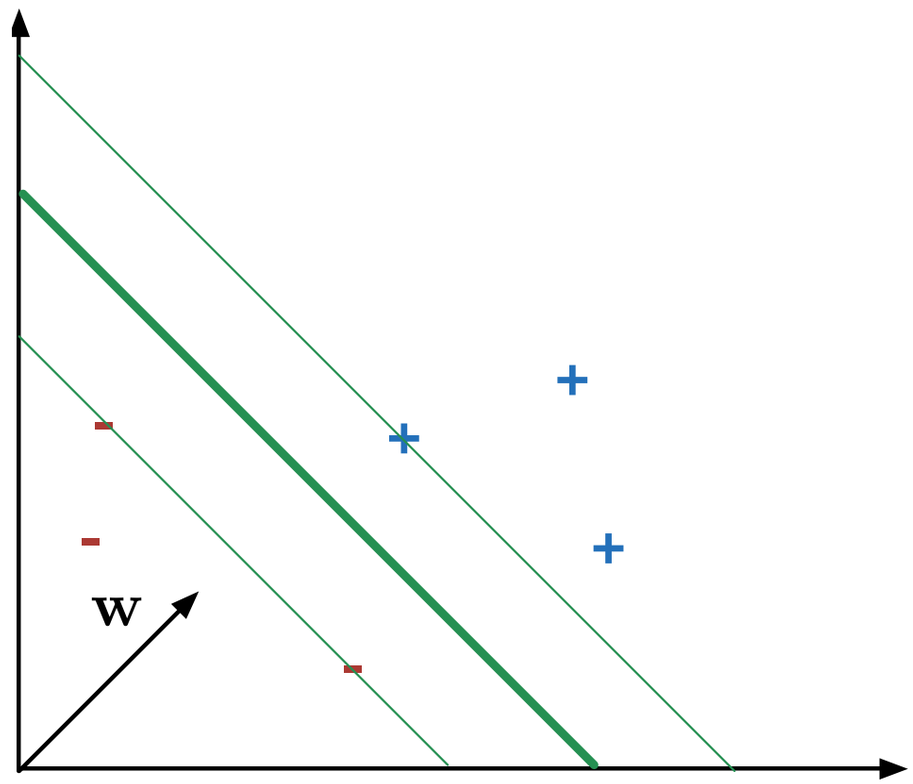


Functional and Geometric Margin

- Our first goal will be to reduce the possible solutions only to those that do not make any error. This will lead to the definition of the *functional margin.*



- Then we can focus on the problem of forcing the solution to maximise the distance from the nearest positive/ negative examples. This will lead to the definition of the *geometric margin.*



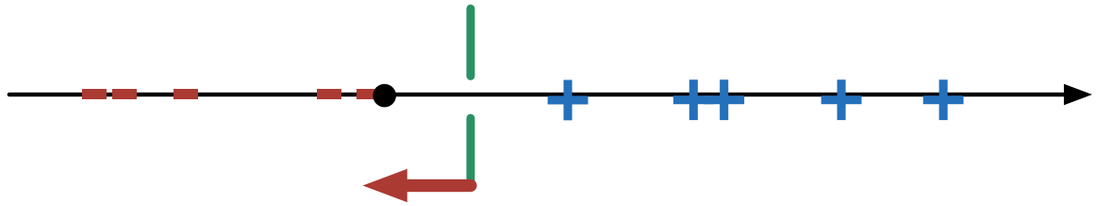
Functional margin

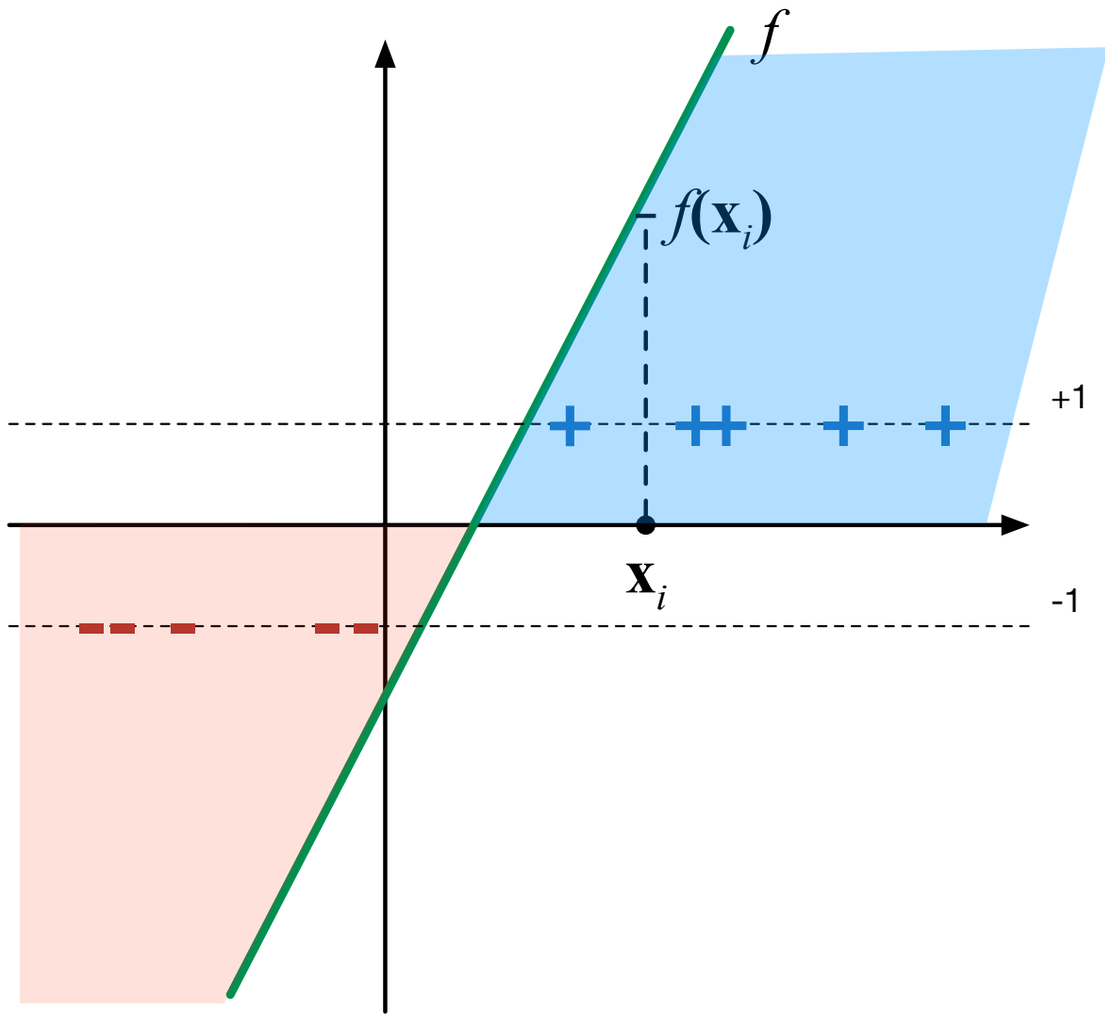
Let us define $f(\mathbf{x}_i) = \mathbf{w} \cdot \mathbf{x}_i - t$ to be the value of the hyperplane at point $\mathbf{x}_i$ and let us use +1, -1 as our labels for positive and negative examples.

Note that we can then use

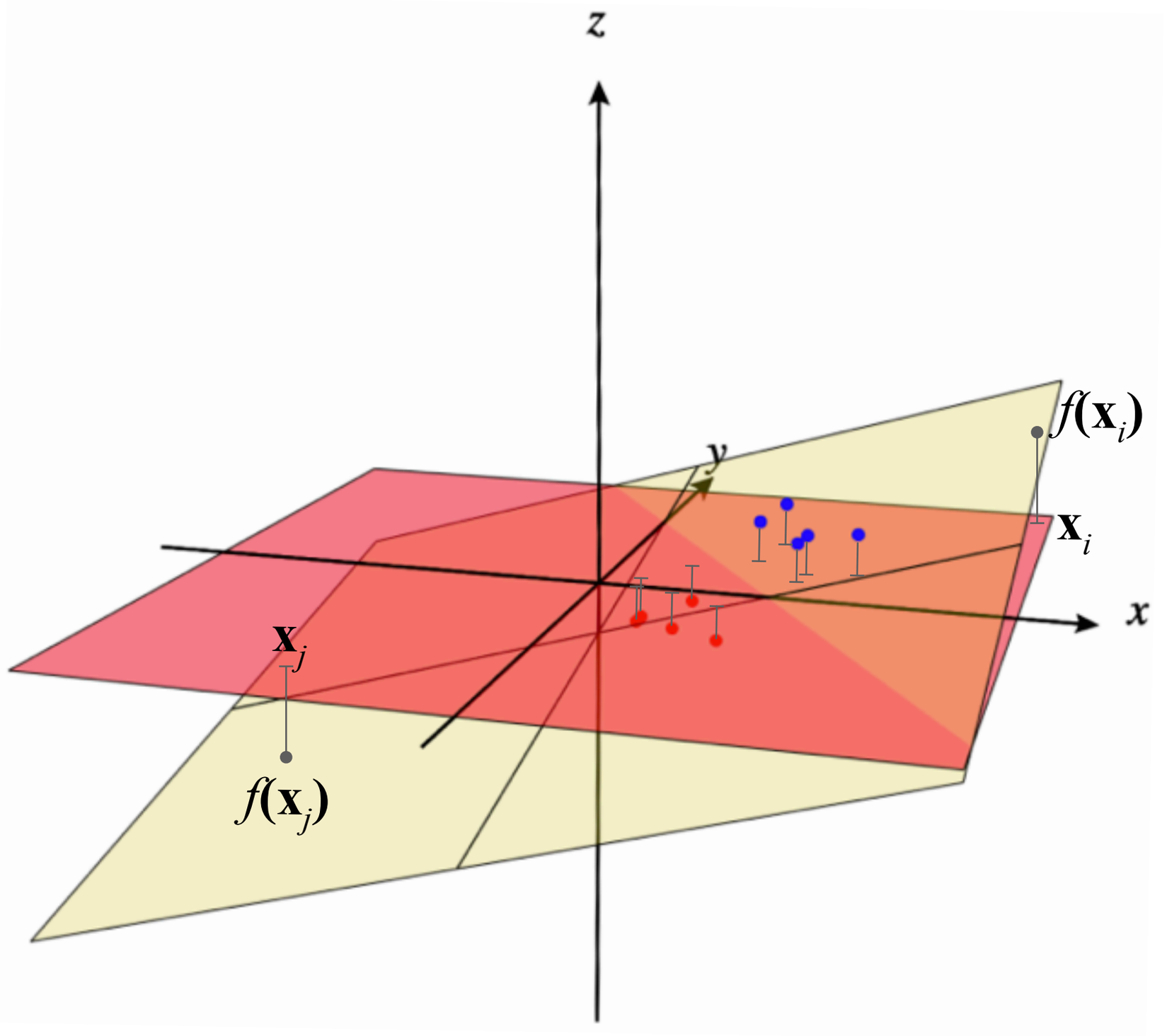
$$f(\mathbf{x}_i) > 0$$

as our decision rule to discriminate between classes.





$f(\mathbf{x}_i)$ example (3D)

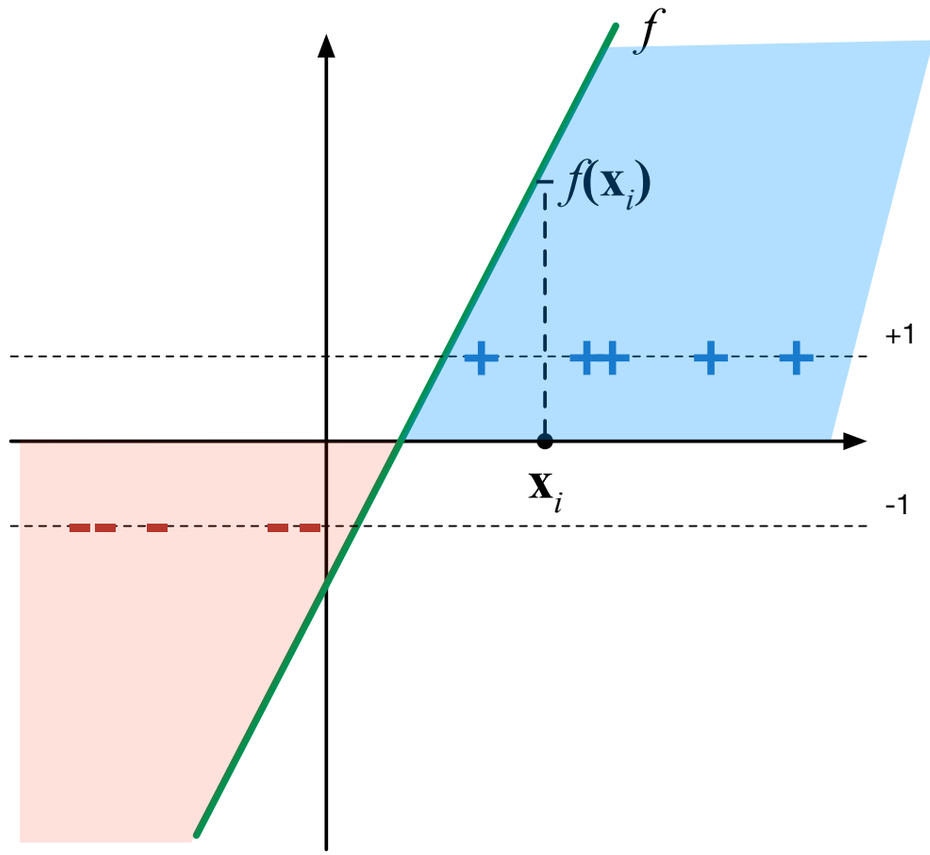


Functional margin

Let us define the *functional margin* of example $\mathbf{x}_i$ w.r.t. the hyperplane determined by $\mathbf{w}$ and t as the quantity:

$$\mu(\mathbf{x}_i) = y_i(\mathbf{w} \cdot \mathbf{x}_i - t) = y_i f(\mathbf{x}_i)$$

Note that $\mu(\mathbf{x}_i) > 0$ iff x is correctly classified.

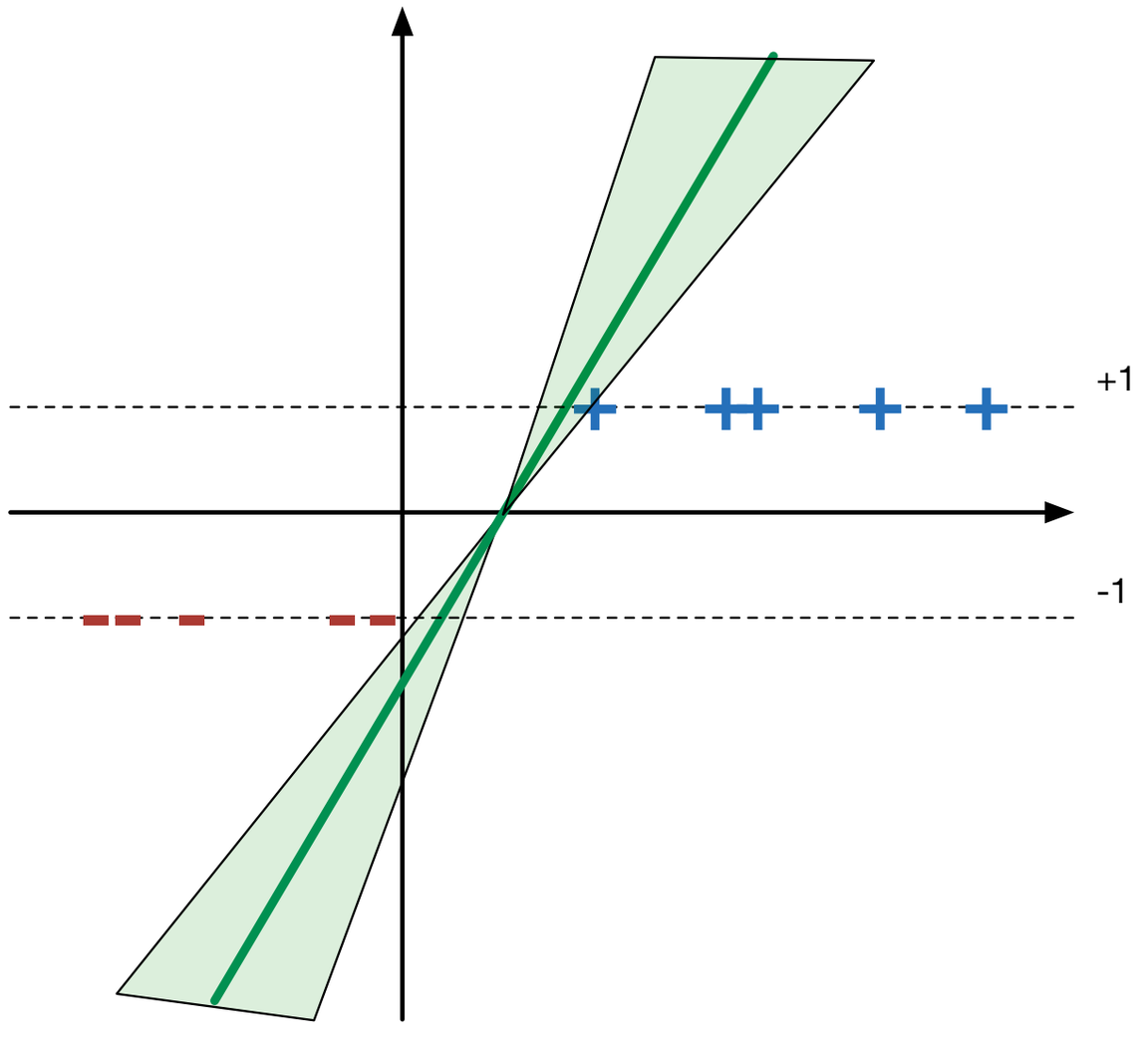


Functional margin

In our solution we will then want a functional margin larger than 0 for each example, i.e.:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - t) > 0$$

If we rescale $\mathbf{w}$ and t we just change the slope of the hyperplane without changing the decision surface.



Support vectors

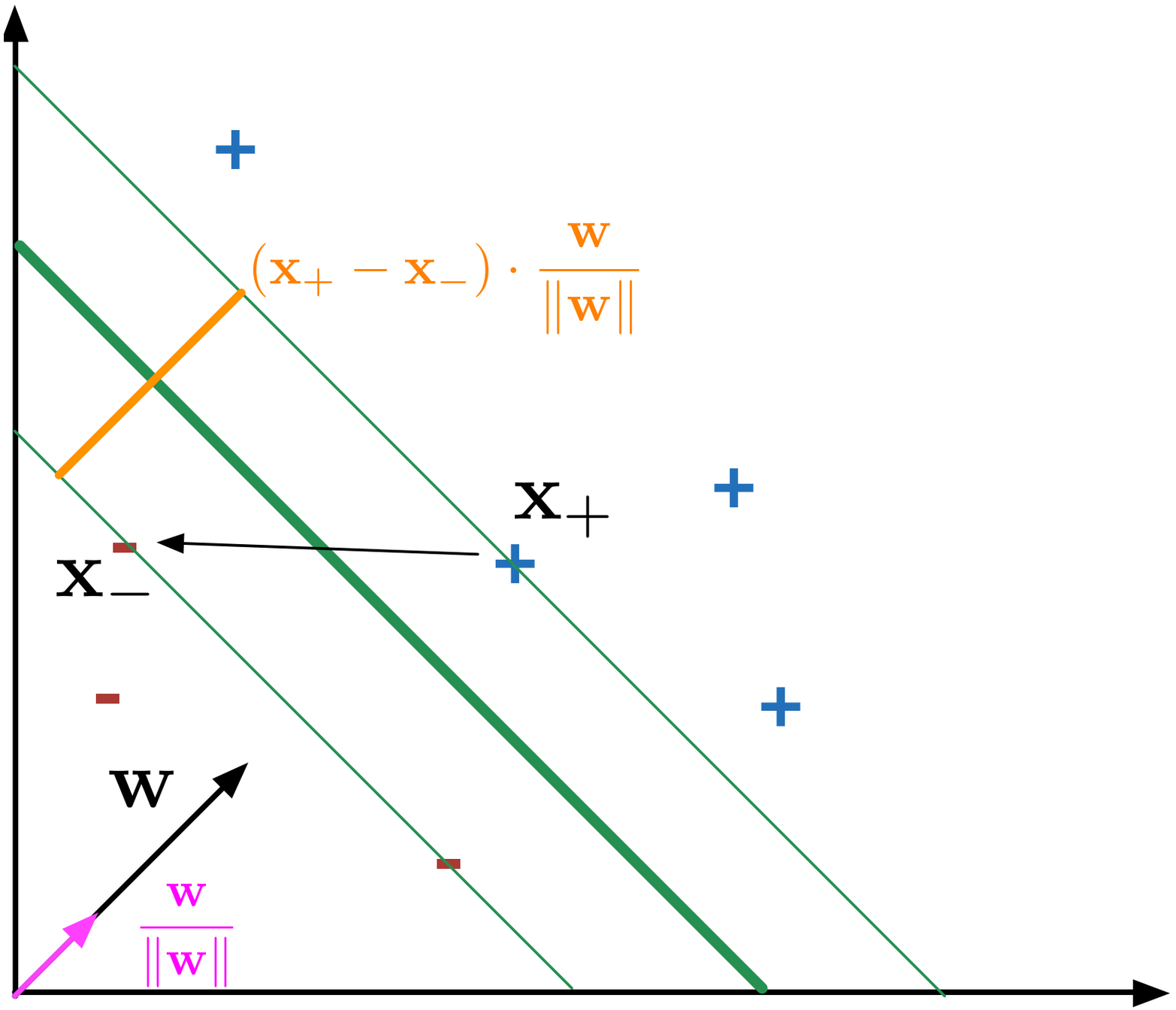
We are then free to require that for each example in the dataset it holds:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - t) \geq 1$$

and also that strict equality holds **for the examples on the boundary**: $y_i(\mathbf{w} \cdot \mathbf{x}_i - t) = 1$

Examples on the boundary have a special name in the SVM theory: they are called the **support vectors**.

Geometric margin



(w_0, w_1) is orthogonal to the discriminating line (hyperplane)

The *discriminating line* is at the intersection of the plane with $z = 0$.

The plane has equation:

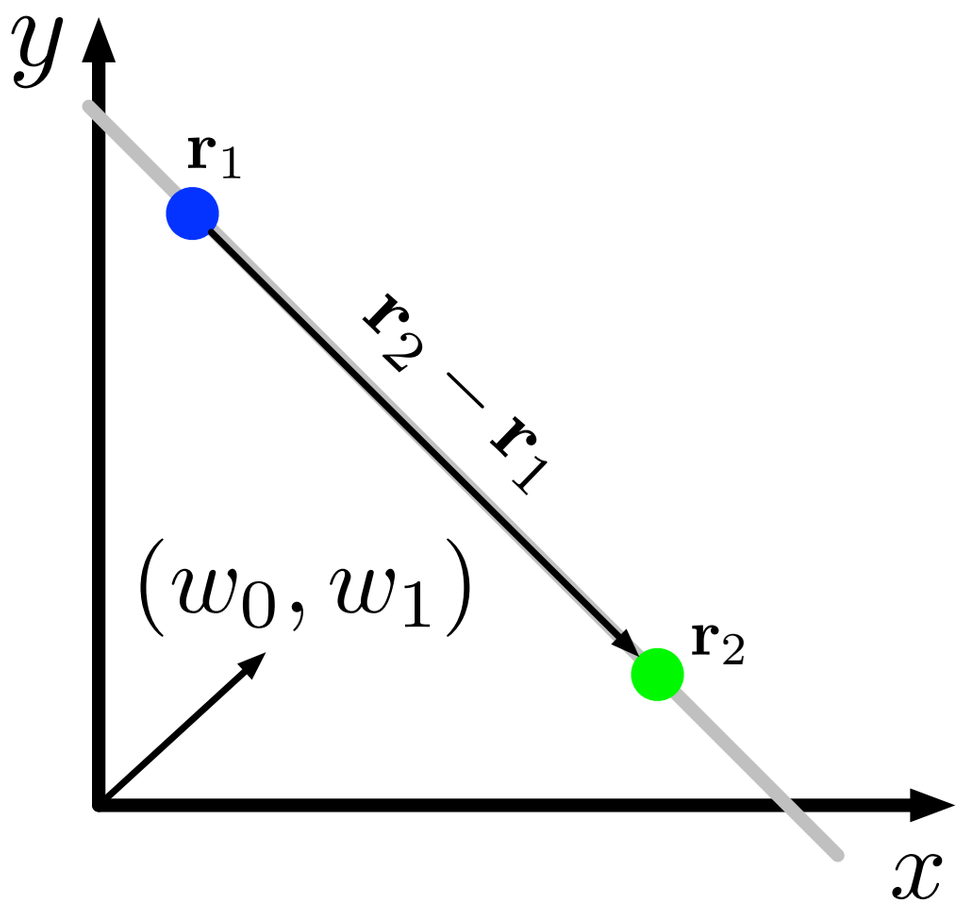
$$w_0 x + w_1 y + w_2 z + d = 0$$

The line has then equation:

$$w_0 x + w_1 y + d = 0$$

For any two points r1 and r2 on the *line* it holds:

$$(w_0, w_1) \cdot (r_1 - r_2) = 0$$



(w_0, w_1) is orthogonal to the discriminating line (hyperplane)

From the line equation:

$$w_0 x + w_1 y + d = 0$$

It follows:

$$r_1 = \left(x_1, -\frac{w_0 x_1 + d}{w_1} \right),\ r_2 = \left(x_2, -\frac{w_0 x_2 + d}{w_1} \right)$$

implying:

$$(w_0, w_1) \cdot (r_2 - r_1) = (w_0, w_1) \cdot \left(x_2 - x_1, -\frac{w_0 x_2 + d}{w_1} + \frac{w_0 x_1 + d}{w_1} \right)$$

$$= w_0(x_2 - x_1) + w_1 \left(-\frac{w_0 x_2 - w_0 x_1}{w_1} \right) = 0$$

SVM: finding the hyperplane maximising the margin

The margin size can be then evaluated as:

$$\mu = (\mathbf{x}_+ - \mathbf{x}_-) \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|}$$
$$= \frac{\mathbf{x}_+ \cdot \mathbf{w}}{\|\mathbf{w}\|} - \frac{\mathbf{x}_- \cdot \mathbf{w}}{\|\mathbf{w}\|}$$

For x_+ and x_-, being examples on the boundary, it holds that:

$$\mathbf{x}_+ \cdot \mathbf{w} - t = 1 \Rightarrow \mathbf{x}_+ \cdot \mathbf{w} = 1 + t$$
$$-(\mathbf{x}_- \cdot \mathbf{w} - t) = 1 \Rightarrow \mathbf{x}_- \cdot \mathbf{w} = t - 1$$

implying:

$$\mu = \frac{1+t}{\|\mathbf{w}\|} - \frac{t-1}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$$

SVM: optimisation problem

To maximise $1/||\mathbf{w}||$ is equivalent to minimise $||\mathbf{w}||$ which, in turn, is equivalent to minimise $||\mathbf{w}||^2$. The SVM optimisation problem can be then formulated as:

$$\begin{aligned} \underset{\mathbf{w},t}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to} \quad & y_i(\mathbf{w}\cdot\mathbf{x}_i - t) \geq 1; \quad 0 \leq i \leq n \end{aligned}$$

Dual formulations

Let us consider the optimisation problem:

$$\begin{aligned} \text{minimize}_x \quad & f_0(x) & \\ \text{subject to} \quad & f_i(x) \leq 0, & i = 1, \ldots, m \\ & g_i(x) = 0, & i = 1, \ldots, p \end{aligned}$$

The Lagrange dual function is, by definition:

$$\begin{aligned} g(\alpha, \nu) &= \inf_x \Lambda(x, \alpha, \nu) \\ &= \inf_x \left(f_0(x) + \sum_{i=1}^{m} \alpha_i f_i(x) + \sum_{i=1}^{p} \nu_i g_i(x) \right) \end{aligned}$$

Duality

Duality can be thought of as “an organised way to form highly non trivial bounds on optimisation problems” (Stephen Boyd).

In convex problems the bound is (usually) strict and maximising the bound (which is the dual formulation of the problem) leads to the same solution as minimising the original function (the primal formulation). In such case we have **strong duality**.

For strong duality a set of conditions known as the **Karush-Kuhn-Tucker (KKT) conditions** needs to hold.

SVM dual problem

In the SVM case, we start by adding multipliers α_i for each inequality constraint to obtain the Lagrange function (with the constraint $\alpha_i \geq 0$):

$$
\begin{aligned}
\Lambda(\mathbf{w}, t, \alpha_1, \ldots, \alpha_n) &= \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \alpha_i (y_i(\mathbf{w}\cdot\mathbf{x}_i - t) - 1) \\
&= \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \alpha_i y_i (\mathbf{w}\cdot\mathbf{x}_i) + \sum_{i=1}^{n} \alpha_i y_i t + \sum_{i=1}^{n} \alpha_i \\
&= \frac{1}{2}\mathbf{w}\cdot\mathbf{w} - \mathbf{w}\cdot\left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i\right) + t\left(\sum_{i=1}^{n} \alpha_i y_i\right) + \sum_{i=1}^{n} \alpha_i
\end{aligned}
$$

SVM dual problem

$$\Lambda(\mathbf{w}, t, \alpha_1, \ldots, \alpha_n) = \frac{1}{2}\mathbf{w}\cdot\mathbf{w} - \mathbf{w}\cdot\left(\sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i\right) + t\left(\sum_{i=1}^{n}\alpha_i y_i\right) + \sum_{i=1}^{n}\alpha_i$$

To write the dual formulation we still have to take the infimum (w.r.t. $\mathbf{x}$ and t) of the Lagrangian. To find the infimum, we start by calculating the partial derivatives of the Lagrangian.

$$\frac{\partial\Lambda}{\partial t} = \sum_i \alpha_i y_i \qquad \frac{\partial\Lambda}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i$$

yielding (by setting them equal to zero):

$$\sum_i \alpha_i y_i = 0 \qquad \mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

SVM dual problem

By substituting back into the Lagrangian, we get the formulation of the dual function:

$$g(\alpha) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i\mathbf{x}_j + \sum_{i=1}^{n}\alpha_i$$

which, as mentioned, is a general lower bound on the solution parametrised by α. By maximising it, we get the dual formulation of the SVM problem:

$$\begin{aligned} \text{maximize}_\alpha \quad & -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i\mathbf{x}_j + \sum_{i=1}^{n}\alpha_i \\ \text{subject to} \quad & \alpha_i \geq 0 \text{ for } i = 1,\ldots,n, \text{ and } \sum_i y_i\alpha_i = 0 \end{aligned}$$

Finding solutions to the SVM problem

How can one find a solution to the primal problem?

Convex optimization (numerical algorithms).

How can one find a solution to the dual problem?

Convex optimization (numerical algorithms).

Why, then, bother to formulate the dual problem?

Advantages of the dual formulation

- ✦ We learn something about the solution:
  - the solution vector **w** is a linear combination of the support vectors;
  - positive Lagrange multipliers are associated with support vectors (implying that non support vectors have null Lagrange multipliers);
  - both learning and classification can be done in terms of dot products of support vectors;
- ✦ In the dual formulation SVM can be used to learn non-linear concepts by using the kernel trick;

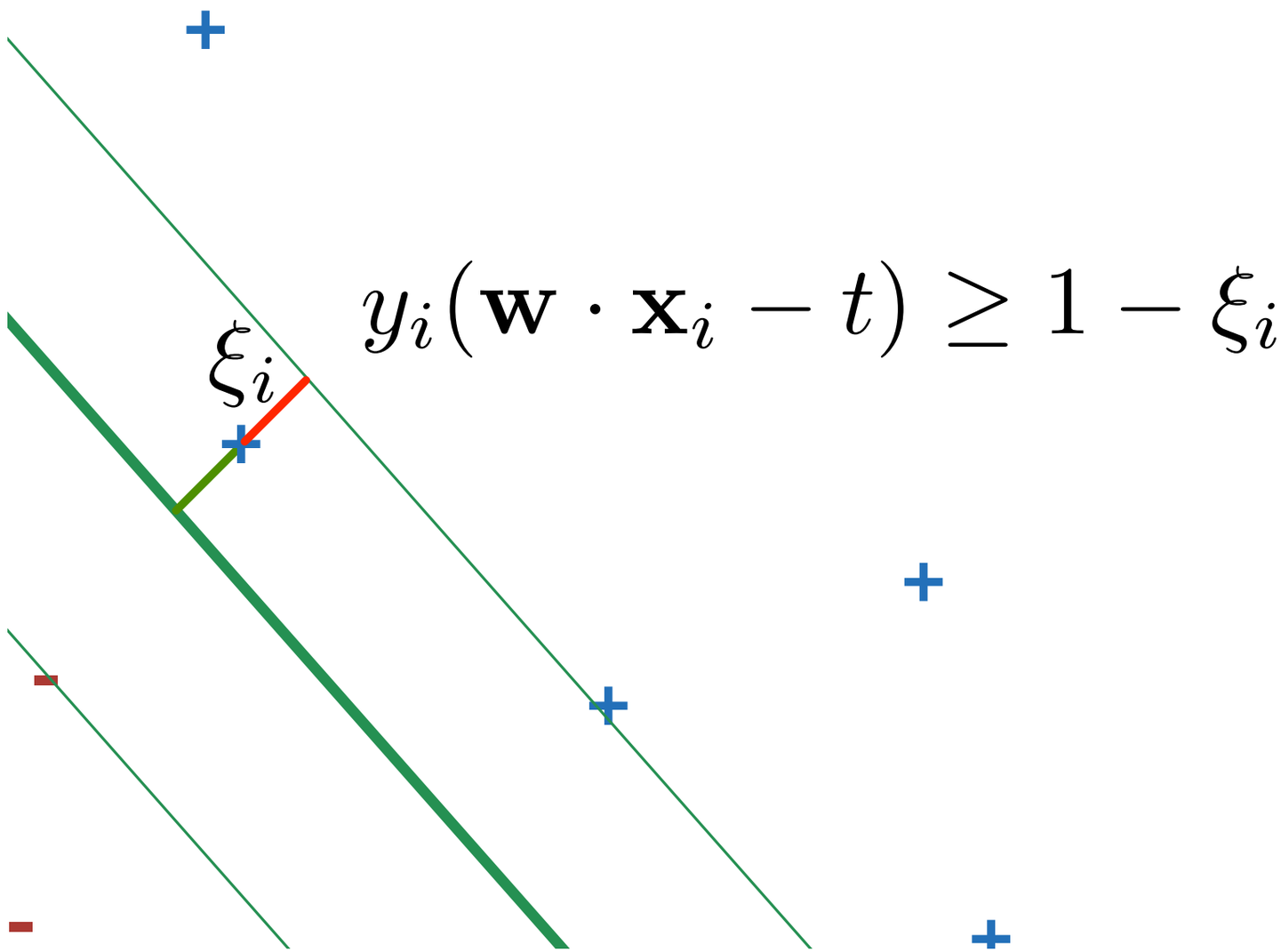
Positive Lagrange multipliers are associated with support vectors

Among the previous observations, the only one that still needs a justification is the assertion that *positive Lagrange multipliers are associated with support vectors.*

This property follows from the KKT condition:

$$\alpha_i(y_i(\mathbf{w}\cdot\mathbf{x}_i - t) - 1) = 0$$

Allowing margin errors



Allowing margin errors

The idea is to introduce slack variables, one for each example, which allow some of them to be inside the margin or even at the wrong side of the decision boundary.

$$\begin{aligned}
\underset{\mathbf{w},t,\xi}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\
\text{subject to} \quad & y_i(\mathbf{w}\cdot\mathbf{x}_i - t) \geq 1 - \xi_i; \qquad 1 \leq i \leq n \\
& \xi_i \geq 0; \qquad 1 \leq i \leq n
\end{aligned}$$

Slack variables relax the constraints, but are penalised in the objective: to find the optimal solution the solver needs to struck a good balance between the maximisation of the margin and the minimisation of the number (and magnitude) of the slack variables.

Allowing margin errors

$$
\begin{aligned}
\underset{\mathbf{w},t,\xi}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\
\text{subject to} \quad & y_i(\mathbf{w}\cdot\mathbf{x}_i - t) \geq 1 - \xi_i; \qquad 1 \leq i \leq n \\
& \xi_i \geq 0; \qquad 1 \leq i \leq n
\end{aligned}
$$

C is a user-defined parameter trading off margin maximisation against slack variable minimisation: a high value of C means that margin errors incur a high penalty, while a low value permits more margin errors (possibly including misclassifications) in order to achieve a large margin.

Allowing margin errors

$$\begin{aligned}
\underset{\mathbf{w},t,\xi}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\
\text{subject to} \quad & y_i(\mathbf{w}\cdot\mathbf{x}_i - t) \geq 1 - \xi_i; \qquad 1 \leq i \leq n \\
& \xi_i \geq 0; \qquad 1 \leq i \leq n
\end{aligned}$$

If we allow more margin errors we need fewer support vectors, hence C controls to some extent the ‘complexity’ of the SVM and hence is often referred to as the complexity parameter.

Dual formulation

$$\begin{aligned}
\underset{\mathbf{w},t,\xi}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\
\text{subject to} \quad & y_i(\mathbf{w}\cdot\mathbf{x}_i - t) \geq 1 - \xi_i; \qquad 1 \leq i \leq n \\
& \xi_i \geq 0; \qquad 1 \leq i \leq n
\end{aligned}$$

The Lagrange function is:

$$\begin{aligned}
\Lambda(\mathbf{w}, t, \xi_i, \alpha_i, \beta_i) &= \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i(y_i(\mathbf{w}\cdot\mathbf{x}_i - t) - (1-\xi_i)) - \sum_{i=1}^{n}\beta_i\xi_i \\
&= \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n}C\xi_i - \sum_{i=1}^{n}\alpha_i(y_i(\mathbf{w}\cdot\mathbf{x}_i - t) - 1) - \sum_{i=1}^{n}\alpha_i\xi_i - \sum_{i=1}^{n}\beta_i\xi_i \\
&= \Lambda(\mathbf{w}, t, \alpha_i) + \sum_{i=1}^{n}C\xi_i - \sum_{i=1}^{n}\alpha_i\xi_i - \sum_{i=1}^{n}\beta_i\xi_i \\
&= \Lambda(\mathbf{w}, t, \alpha_i) + \sum_{i=1}^{n}\xi_i(C - \alpha_i - \beta_i)
\end{aligned}$$

Dual formulation

By setting the derivative of the ξ_i variables to zero we obtain C - α_i - $\beta_i = 0$, implying:

$$\Lambda(\mathbf{w}, t, \xi_i, \alpha_i, \beta_i) = \Lambda(\mathbf{w}, t, \alpha_i)$$

Which allows us to write the dual formulation of the soft margin SVM problem as:

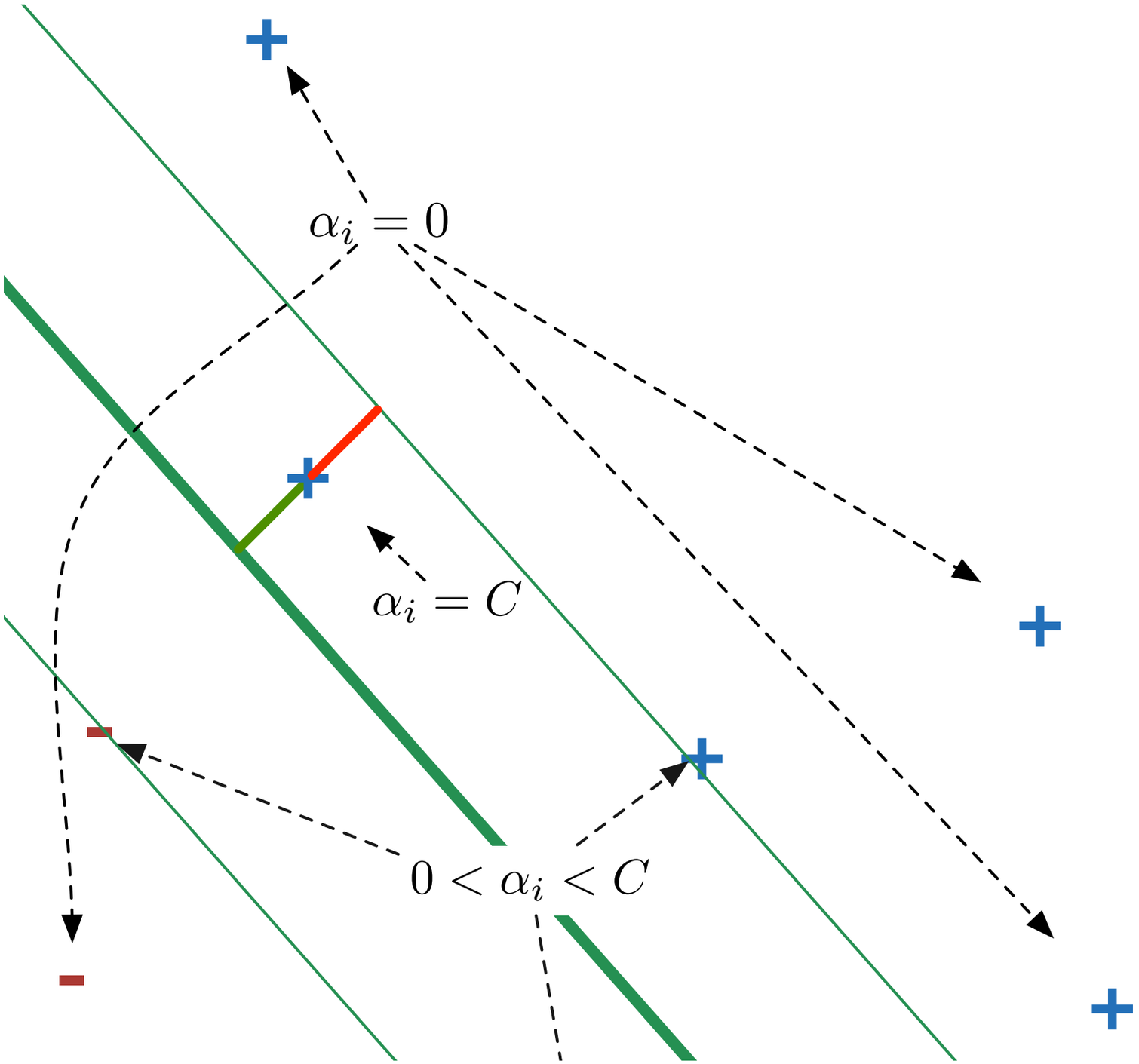
$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j + \sum_{i=1}^{n}\alpha_i \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C; \quad i = 1, \ldots, n \\ & \sum_{i=1}^{n} y_i \alpha_i = 0 \end{aligned}$$

Dual formulation

Also, one of the KKT conditions states that $\beta_i \xi_i = 0$ implying that if an example is a *margin error* (i.e., has $\xi_i > 0$) then $\beta_i = 0$, which in turns implies $\alpha_i = C$ (see previous slide). From these considerations (and from everything we already know about support vectors) it follows that examples for which:

$\alpha_i = 0$	are examples outside (or on) the margin;
$0 < \alpha_i < C$	are support vectors
$\alpha_i = C$	are margin errors

Slack variables interpretation



Kernels

The kernel trick

The “kernel trick” is an established way to extend (some) linear algorithms to cope with non-linear hypotheses spaces.

The main idea is based on the fact that a linear decision surface on a transformed space may correspond to a non-linear decision surface on the original feature space.

The kernel trick

Example: consider the mapping

$$\phi(\mathbf{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2, c)$$

and the form of the classification function of the svm:

$$\hat{c}(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - t)$$

and how it changes after substituting the dot product with a kernel based on the above mapping:

$$\begin{aligned}
\hat{c}(\mathbf{x}) &= \text{sign}(K(\mathbf{w}, \mathbf{x}) - t) = \text{sign}(\phi(\mathbf{w}) \cdot \phi(\mathbf{x}) - t) \\
&= \text{sign}((w_1^2, \sqrt{2}w_1w_2, w_2^2, c) \cdot (x_1^2, \sqrt{2}x_1x_2, x_2^2, c) - t) \\
&= \text{sign}(w_1^2x_1^2 + 2w_1w_2x_1x_2 + w_2^2x_2^2 + c^2 - t)
\end{aligned}$$

Kernel functions

A kernel function is a function $K: \mathbb{V} \times \mathbb{V} \rightarrow \mathbb{R}$ for which it holds that there exists a mapping $\phi: \mathbb{V} \rightarrow \mathbb{F}$ (with $\mathbb{F}$ being a Hilbert space, i.e., a complete vector space equipped with an inner product), such that:

$$K(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$$

i.e., a kernel function computes an inner product of $\mathbf{x}$ and $\mathbf{y}$ after mapping them into a different (possibly very high dimensional) Hilbert space.

Inner products

An inner product is a generalisation of a dot product to arbitrary vector spaces. It is axiomatically defined as a function that associates each pair of vectors a scalar (let's say a real number, even though it should be defined over arbitrary fields). To be an inner product the function must satisfy:

- symmetry: $\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{y}, \mathbf{x} \rangle$
- linearity in the first argument: $\langle a\mathbf{x}+b\mathbf{y}, \mathbf{z} \rangle = a\langle \mathbf{x}, \mathbf{z} \rangle + b\langle \mathbf{y}, \mathbf{z} \rangle$
- positive-definiteness: $\langle \mathbf{x}, \mathbf{x} \rangle \geq 0; \;\; \langle \mathbf{x}, \mathbf{x} \rangle = 0 \Leftrightarrow \mathbf{x} = 0$

Kernels: motivations

- ✦ representational:
  - extend linear classifiers to cope with non-linear problems;
- ✦ computational:
  - compute a similarity function in a high-dimensional feature space without actually computing the feature vectors;
- ✦ theoretical:
  - conditions exist to establish if a given function is a kernel (to actually prove they hold, can be hard);
  - compositional properties allow one to build new kernels from known ones.

Important kernels

Polinomial Kernel of degree d:

$$K(\mathbf{x},\mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^d \quad \text{or} \quad K(\mathbf{x},\mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^d$$

Notice:

- for $d = 1$ we have the linear (trivial/identity) Kernel
- for $d = 2$ we have the quadratic Kernel (often used in Natural Language Processing problems)
- for a generic d the considered feature space has a dimensionality which is exponential in d.

Important Kernels

Gaussian Kernel:

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\sigma^2}\right)$$

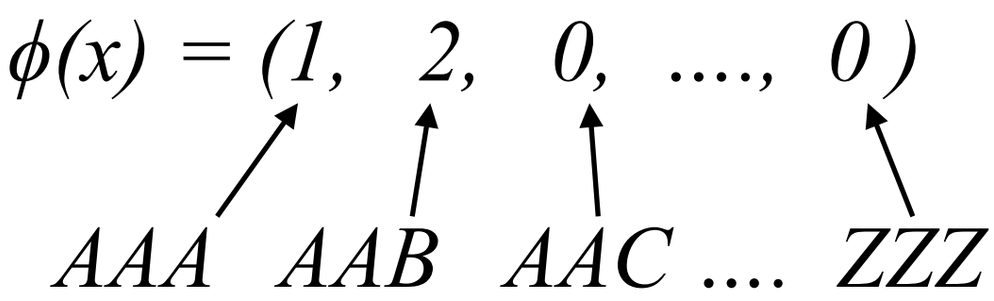
Note:

- the features space has infinite dimensionality
- σ is a parameter usually set by cross validation on an independent validation set

Other kernels

Many other kernels exist. The concept can also be applied to structured data such as strings or trees.

String kernel (often used in biology) example:



The mapping function stores in a vector the number of times each substring of the given length (3 in our example) appear in string x. The kernel is actually computed using a dynamic programming algorithm.

Kernel intuition

Kernels can be thought of similarity functions in the “output” feature space.

In fact, if the vectors are “similar” then they probably point in a similar direction and the dot product is likely to be large.

If the vectors are “dissimilar” then they probably point in different directions and the dot product ought to be small.

Mercer's conditions

Let's say that one has built a function $K(\mathbf{x},\mathbf{y})$ which implements a similarity between $\mathbf{x}$ and $\mathbf{y}$. How can he/she be sure that K is a valid kernel? I.e., how can he/she be sure that there exists a Hilbert space H such that K evaluates the inner product in H?

Mercer's conditions:

*A function K is a valid kernel **if** and **only if** for any finite set of points $\{x_1,\ldots,x_m\}$, the matrix $\mathbf{K}$ (defined as $\mathbf{K}_{i,j} = K(\mathbf{x}_i,\mathbf{x}_j)$) is symmetric and positive semi-definite.*

Mercer's conditions: if K is a kernel, then $\mathbf{K}$ is symmetric and $\mathbf{K} \geq 0$

Let us prove the if part of the statement about the Mercer's conditions (the other direction will not be proved here).

We want to prove that if there exists ϕ such that $K(\mathbf{x},\mathbf{y})=\langle\phi(\mathbf{x}), \phi(\mathbf{y})\rangle$, then for any set of points $\{x_1,\ldots,x_n\}$, $\mathbf{K}$ is symmetric and positive semidefinite.

The symmetric part is trivial (by definition of inner product).

Mercer's conditions: if K is a kernel, then $\mathbf{K} \geq 0$

Let us now show that $\mathbf{K} \geq \mathbf{0}$, i.e., that for all $\mathbf{z}$: $\mathbf{z}^T \mathbf{K}\, \mathbf{z} \geq 0$.

$$\mathbf{z}^T \mathbf{K} \mathbf{z} = \sum_i \sum_j z_i z_j \mathbf{K}_{i,j} = \sum_i \sum_j z_i z_j K(\mathbf{x}_i, \mathbf{x}_j) =$$

$$= \sum_i \sum_j z_i z_j \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$$

$$= \sum_i \sum_j \langle z_i \phi(\mathbf{x}_i), z_j \phi(\mathbf{x}_j) \rangle$$

$$= \sum_i \langle z_i \phi(\mathbf{x}_i), \sum_j z_j \phi(\mathbf{x}_j) \rangle$$

$$= \langle \sum_i z_i \phi(\mathbf{x}_i), \sum_j z_j \phi(\mathbf{x}_j) \rangle = \langle \sum_i z_i \phi(\mathbf{x}_i), \sum_i z_i \phi(\mathbf{x}_i) \rangle \geq 0$$

Building new kernels

- $K(\mathbf{x},\mathbf{y}) = K_1(\mathbf{x},\mathbf{y})\, K_2(\mathbf{x},\mathbf{y})$ Roughly equivalent to AND ops
- $K(\mathbf{x},\mathbf{y}) = K_1(\mathbf{x},\mathbf{y}) + K_2(\mathbf{x},\mathbf{y})$ Roughly equivalent to OR ops
- $K(\mathbf{x},\mathbf{y}) = aK_1(\mathbf{x},\mathbf{y})$, *for* $a>0$
- $K(\mathbf{x},\mathbf{y}) = f(\mathbf{x})\, f(\mathbf{y})$, *for any function* f
- …