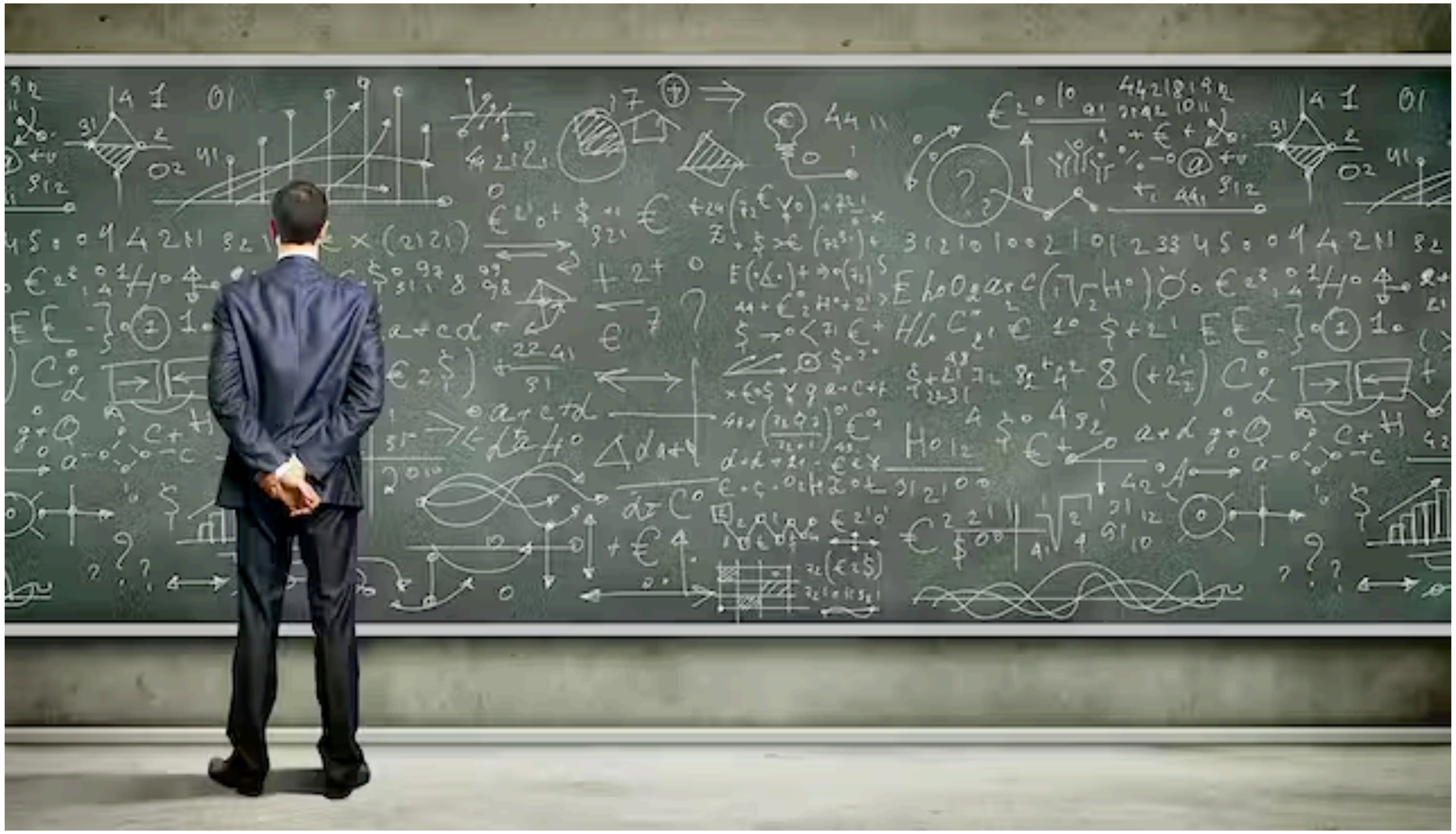


Revision - Classification

Leandro L. Minku

Overview of the Module

Definition and components of supervised learning

Classification approaches and underlying optimisation algorithms

Regression approaches and underlying optimisation algorithms

Foundational theory

Overview of the Module

Definition and components of supervised learning

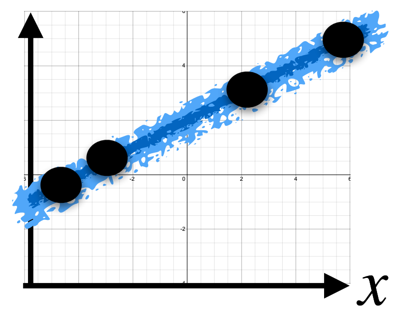
Classification approaches and underlying optimisation algorithms

Regression approaches and underlying optimisation algorithms

Foundational theory

Components of the Supervised Learning Process in View of Noise

$f(x) = 0.5x + 2 + noise$



Unknown Target Function
$f: \mathcal{X} \to \mathcal{Y}$

Unknown Target Distribution
$p(y \mid \mathbf{x})$

Unknown Input Distribution
$p(\mathbf{x})$

Training Examples
$T = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$

$T = \{(4,4), (0,2), (8,6), (2,3)\}$

$T = \{(4.3,4), (0,2.2), (7.9,5.1), (2.3,3.1)\}$

Learning Algorithm
$\mathcal{A}$

Final Hypothesis
$g \approx f$

Hypothesis Set
$\mathcal{H}$

$h(\mathbf{x}) = \mathbf{a}^T\mathbf{x} + b, \quad \forall \mathbf{a} \in R^d, b \in R$

Is Learning Feasible?

$$P\left(\left|E_{in}(g) - E_{out}(g)\right| > \epsilon\right) \leq 2Me^{-2\epsilon^2 N}$$

Assumption: examples in $\mathcal{T}$ are drawn i.i.d. from $p(\mathbf{x}, y)$, and so do any test examples.

Probability of the training error being a "bad" estimation of the generalisation error is smaller than a value that decreases exponentially with the increase of ϵ^2 and N, and increases linearly with the increase of M.

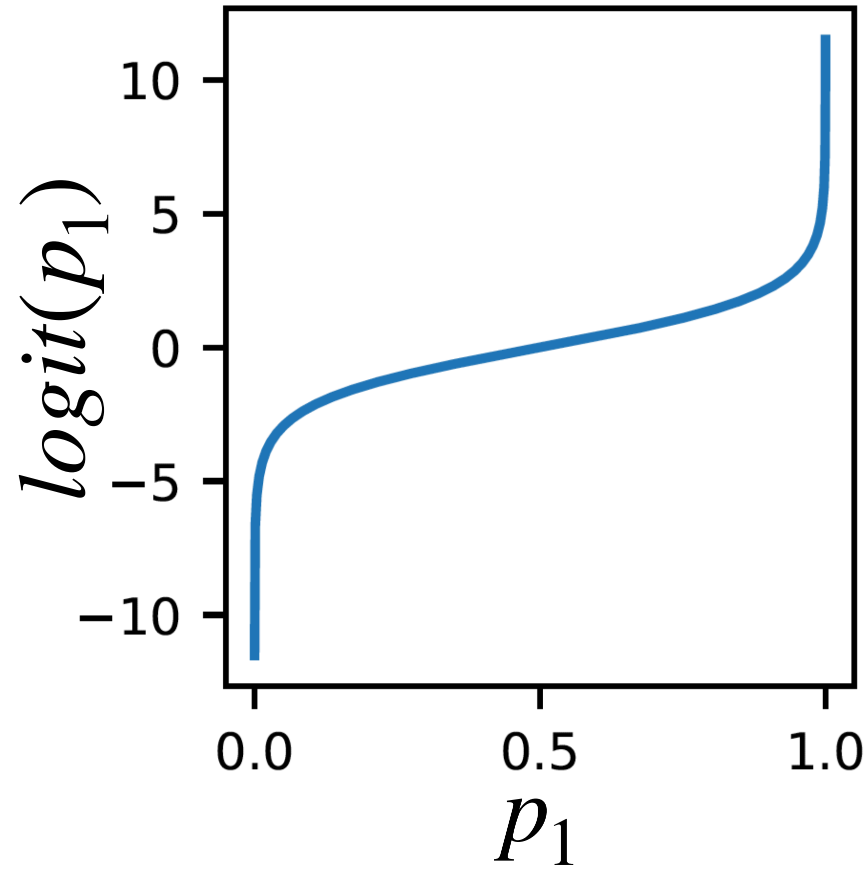
It makes sense to estimate the generalisation error based on the training error to learn a model, but note the effect of M…!

Logistic Regression

- Models logit$(p_1) = \mathbf{w}^T\mathbf{x}$, where

  logit$(p_1) = \ln\left(\frac{p_1}{1 - p_1}\right)$

- Logit enables us to map from [0,1] to [-∞,∞].



Computing the Probabilities p_1 and p_0

- $\text{logit}(p_1) = \mathbf{w}^T\mathbf{x}$
  - $\mathbf{w}^T\mathbf{x} \geq 0 \rightarrow$ class 1
  - $\mathbf{w}^T\mathbf{x} < 0 \rightarrow$ class 0

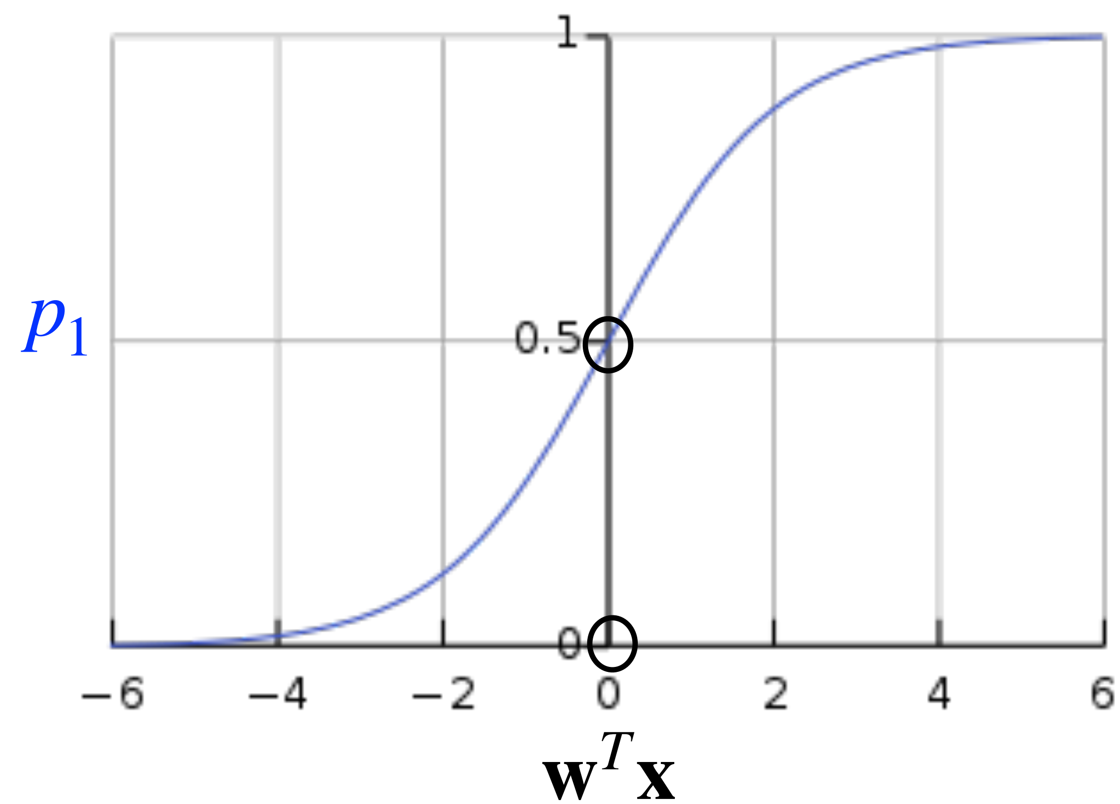
- If we solve $\text{logit}(p_1) = \mathbf{w}^T\mathbf{x}$ for p_1 we get:

$$p_1 = \frac{e^{(\mathbf{w}^T\mathbf{x})}}{1 + e^{(\mathbf{w}^T\mathbf{x})}}$$

$$p_0 = 1 - p_1 = \frac{1}{1 + e^{(\mathbf{w}^T\mathbf{x})}}$$

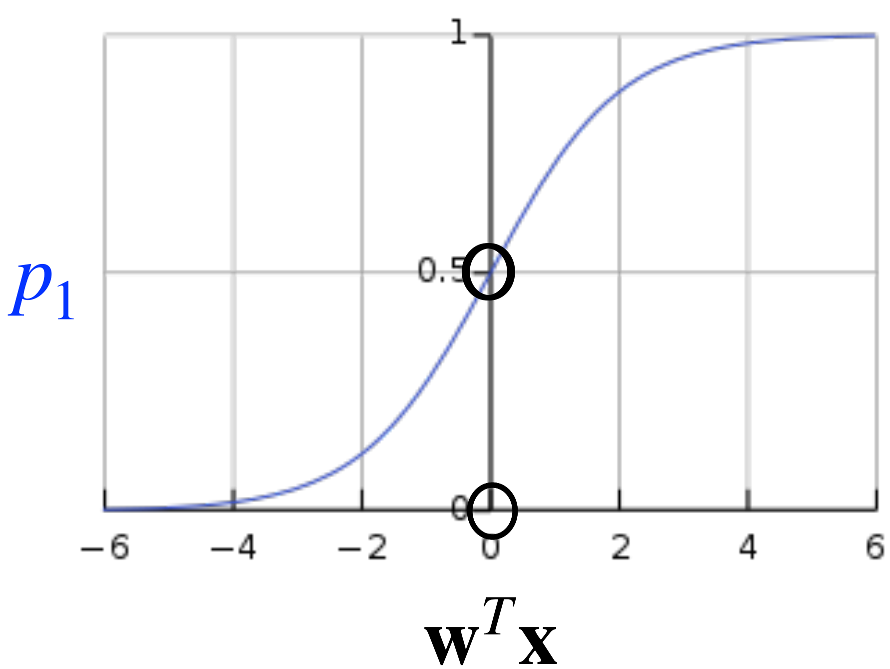
- $p_1 \geq 0.5 \rightarrow$ class 1
- $p_1 < 0.5 \rightarrow$ class 0

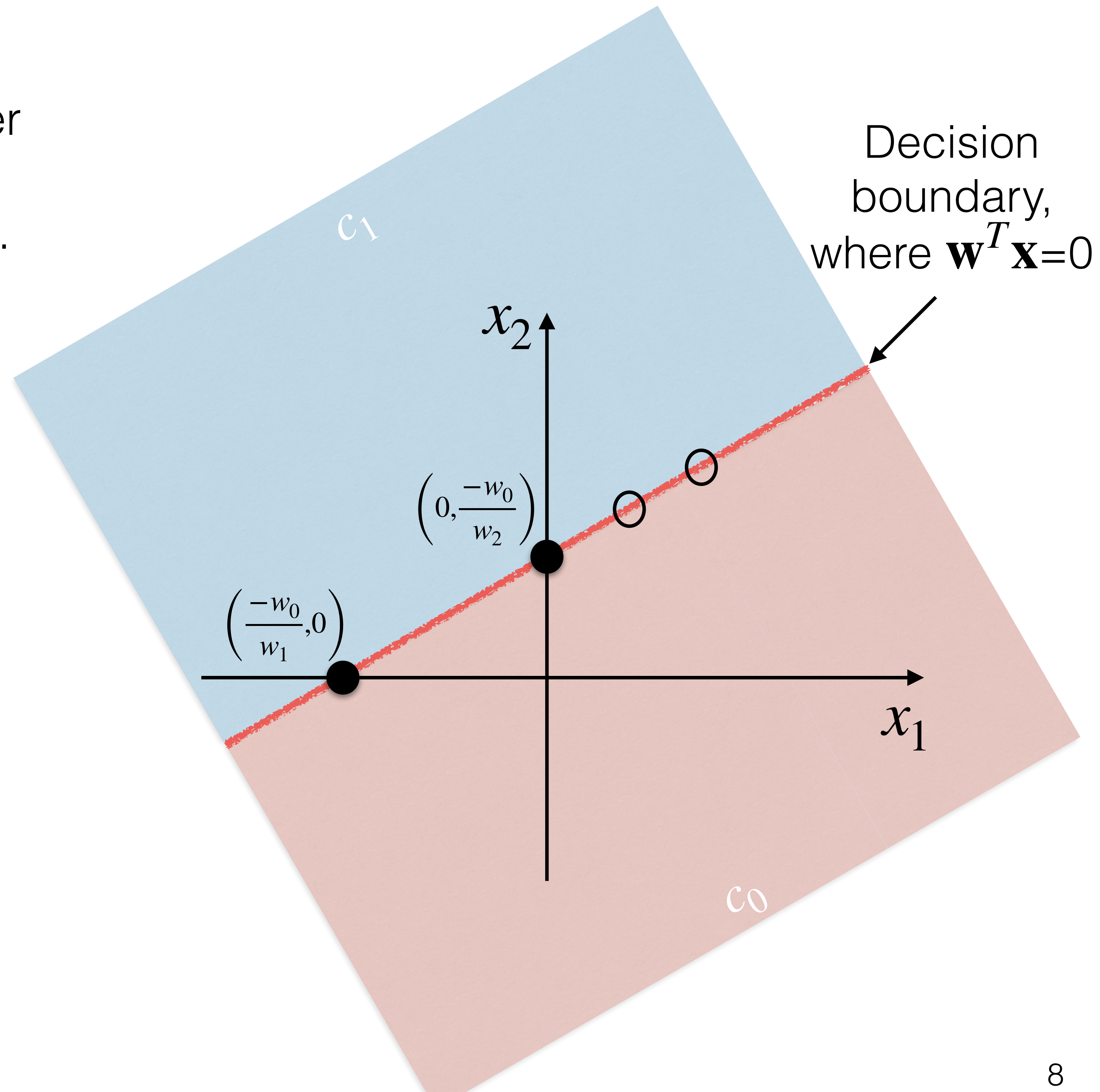
Sigmoid logistic function



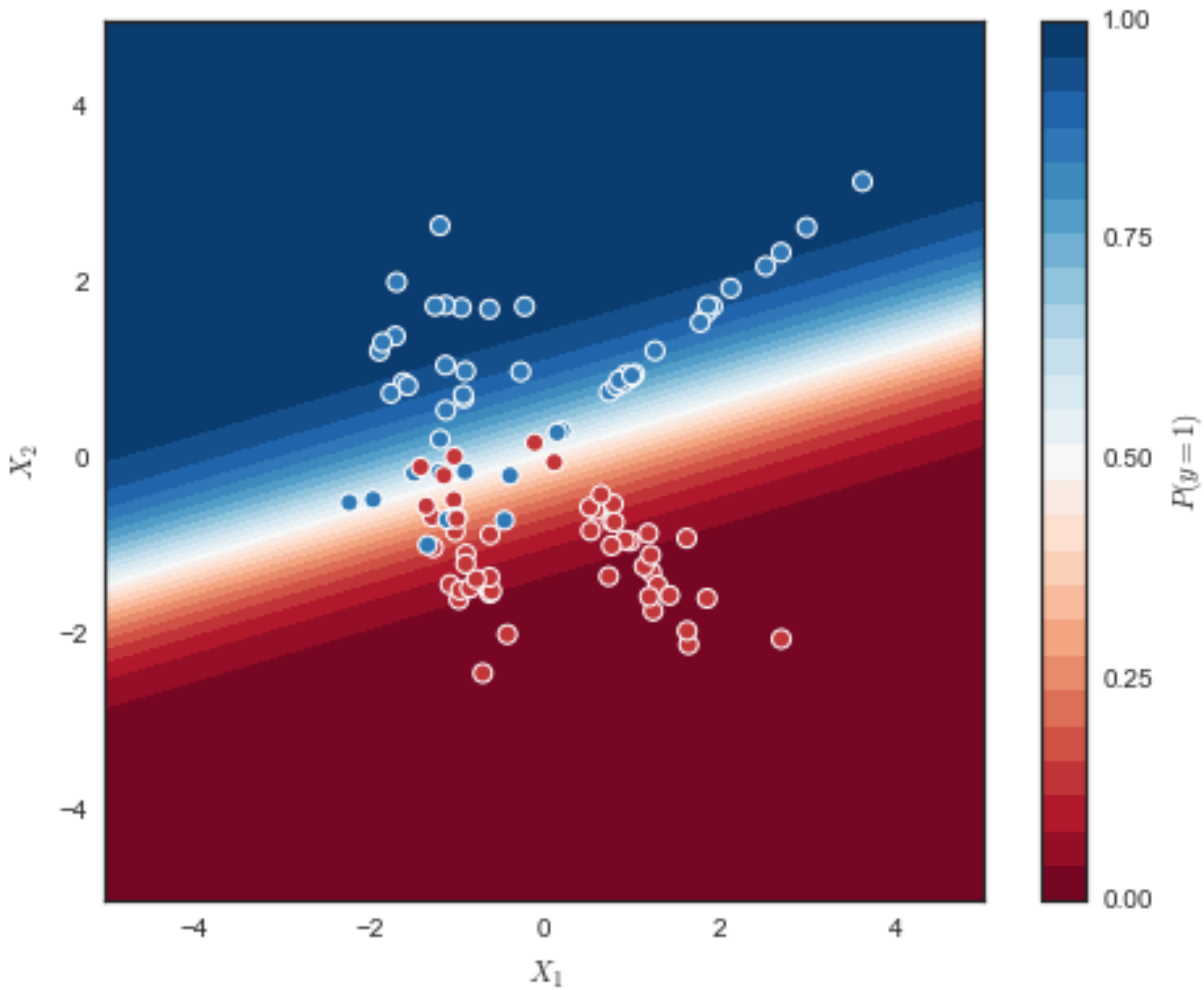
A Linear Classifier

- The larger $|\mathbf{w}^T\mathbf{x}|$, the further away from the decision boundary the example $\mathbf{x}$ is.
- The larger $\mathbf{w}^T\mathbf{x}$, the higher p_1.
- The more negative $\mathbf{w}^T\mathbf{x}$, the smaller the p_1 (and the larger the p_0).





Visualising the Probabilities



Hypothesis Set

- logit$(p_1) = \mathbf{w}^T\mathbf{x}$
  - $\mathbf{w}^T\mathbf{x} \geq 0 \rightarrow$ class 1
  - $\mathbf{w}^T\mathbf{x} < 0 \rightarrow$ class 0
- If we solve logit$(p_1) = \mathbf{w}^T\mathbf{x}$ for p_1 we get:

$$p_1 = \frac{e^{(\mathbf{w}^T\mathbf{x})}}{1 + e^{(\mathbf{w}^T\mathbf{x})}}$$

$$p_0 = 1 - p_1 = \frac{1}{1 + e^{(\mathbf{w}^T\mathbf{x})}}$$

- $p_1 \geq 0.5 \rightarrow$ class 1
- $p_1 < 0.5 \rightarrow$ class 0

$$h(\mathbf{x}) = \begin{cases} 1 & \text{if logit}(p_1) \geq 0 \\ 0 & \text{otherwise} \end{cases}, \quad \forall \mathbf{w} \in R^{d+1}$$

$$h(\mathbf{x}) = \begin{cases} 1 & \text{if } p_1 = p(1 \,|\, \mathbf{x}, \mathbf{w}) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}, \quad \forall \mathbf{w} \in R^{d+1}$$

$$h(\mathbf{x}) = p_1 = p(1 \,|\, \mathbf{x}, \mathbf{w}), \quad \forall \mathbf{w} \in R^{d+1}$$

Maximum Likelihood Estimation

Principle: the most reasonable values for $\mathbf{w}$ are the ones for which the "probability" of the observed examples is largest.

$$\mathscr{L}(\mathbf{w}) = \prod_{i=1}^{N} p(y^{(i)} \,|\, \mathbf{x}, \mathbf{w}) = \prod_{i=1}^{N} p_{y^{(i)}}$$

Cross Entropy Loss

$$E(\mathbf{w}) = -\sum_{i=1}^{N} y^{(i)} \ln p(1 \,|\, \mathbf{x}^{(i)}, \mathbf{w}) + (1 - y^{(i)}) \ln (1 - p(1 \,|\, \mathbf{x}^{(i)}, \mathbf{w}))$$

Cross-entropy is a measure of dissimilarity between two probability distributions.

Here, it is used to measure the dissimilarity between the true (target) distribution $P(y \,|\, \mathbf{x})$ and learned distribution $p(y \,|\, \mathbf{x}, \mathbf{w})$, estimated based on the training examples.

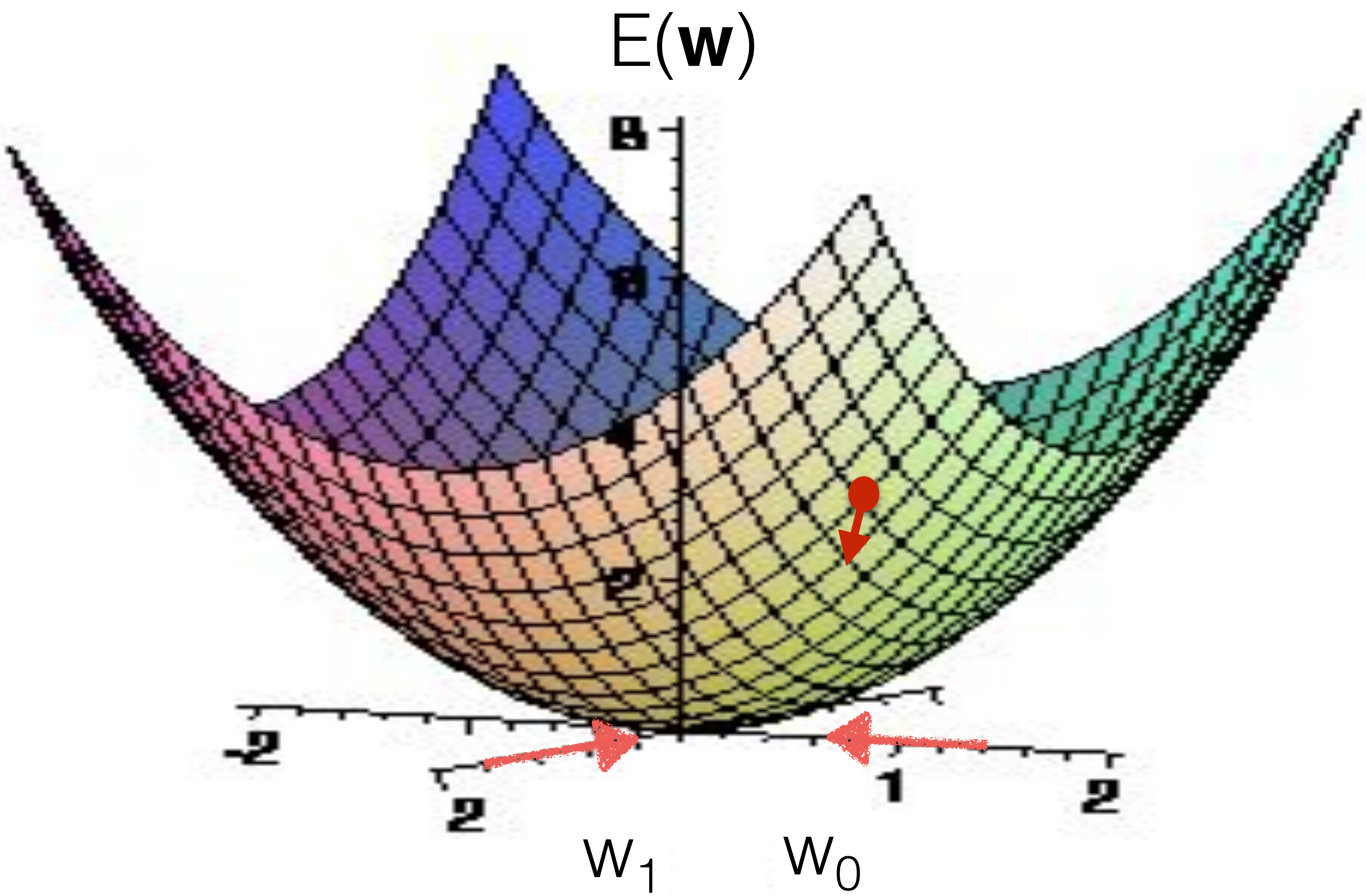
Gradient Descent (Batch Version)

Initialise $\mathbf{w}$ with zeroes or random values near zero.

Repeat for a given number of iterations or until $\nabla E(\mathbf{w})$ is a vector of zeroes:

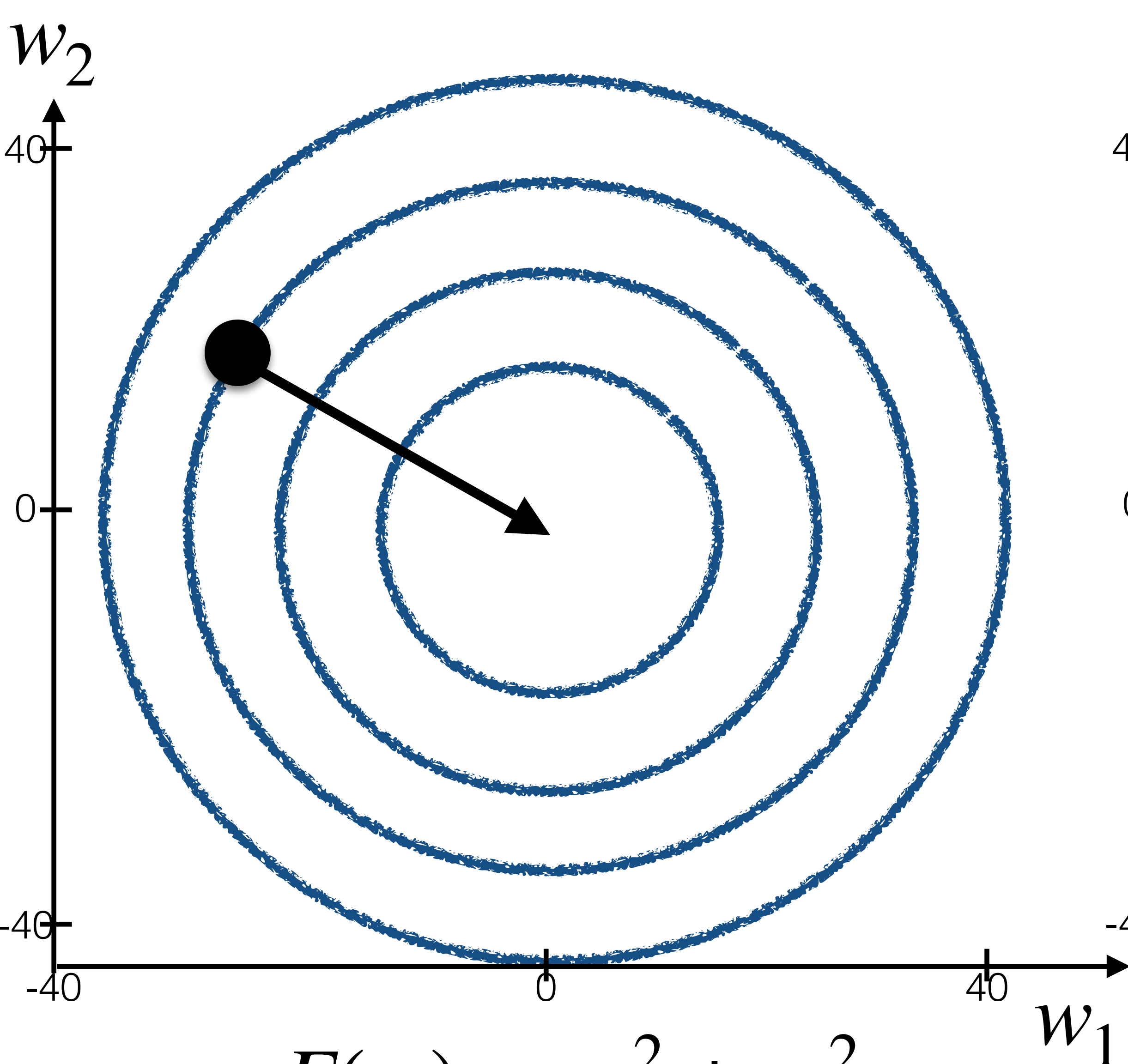
$\mathbf{w} = \mathbf{w} - \eta \nabla E(\mathbf{w})$ where $\eta > 0$ is the learning rate.

Steepest Descent



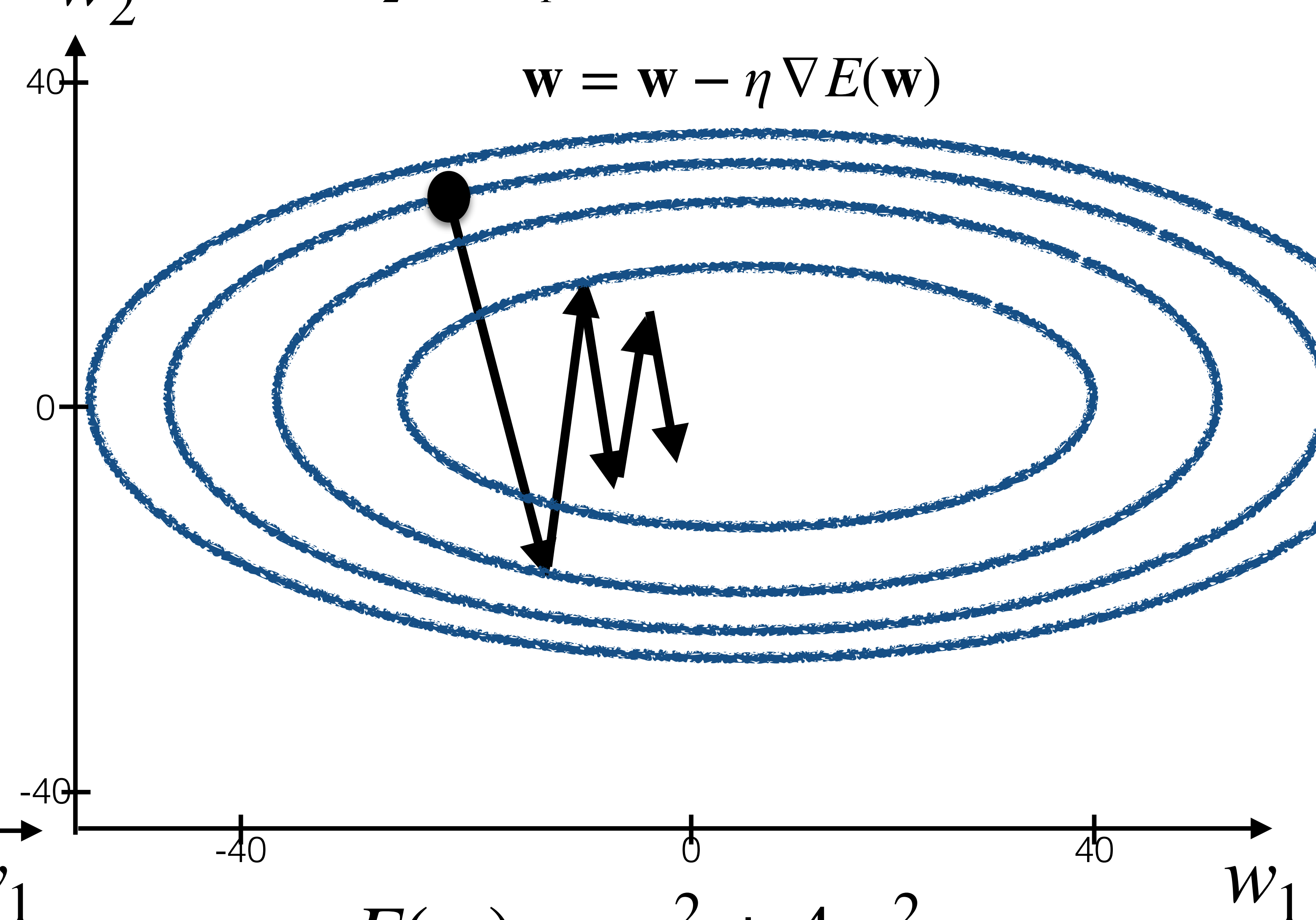
Changes coefficients $\mathbf{w}$ in the direction of the **steepest** descent, i.e., the direction that causes the largest reduction in $E(\mathbf{w})$.

Differential Curvature



$$E(\mathbf{w}) = w_1^2 + w_2^2$$

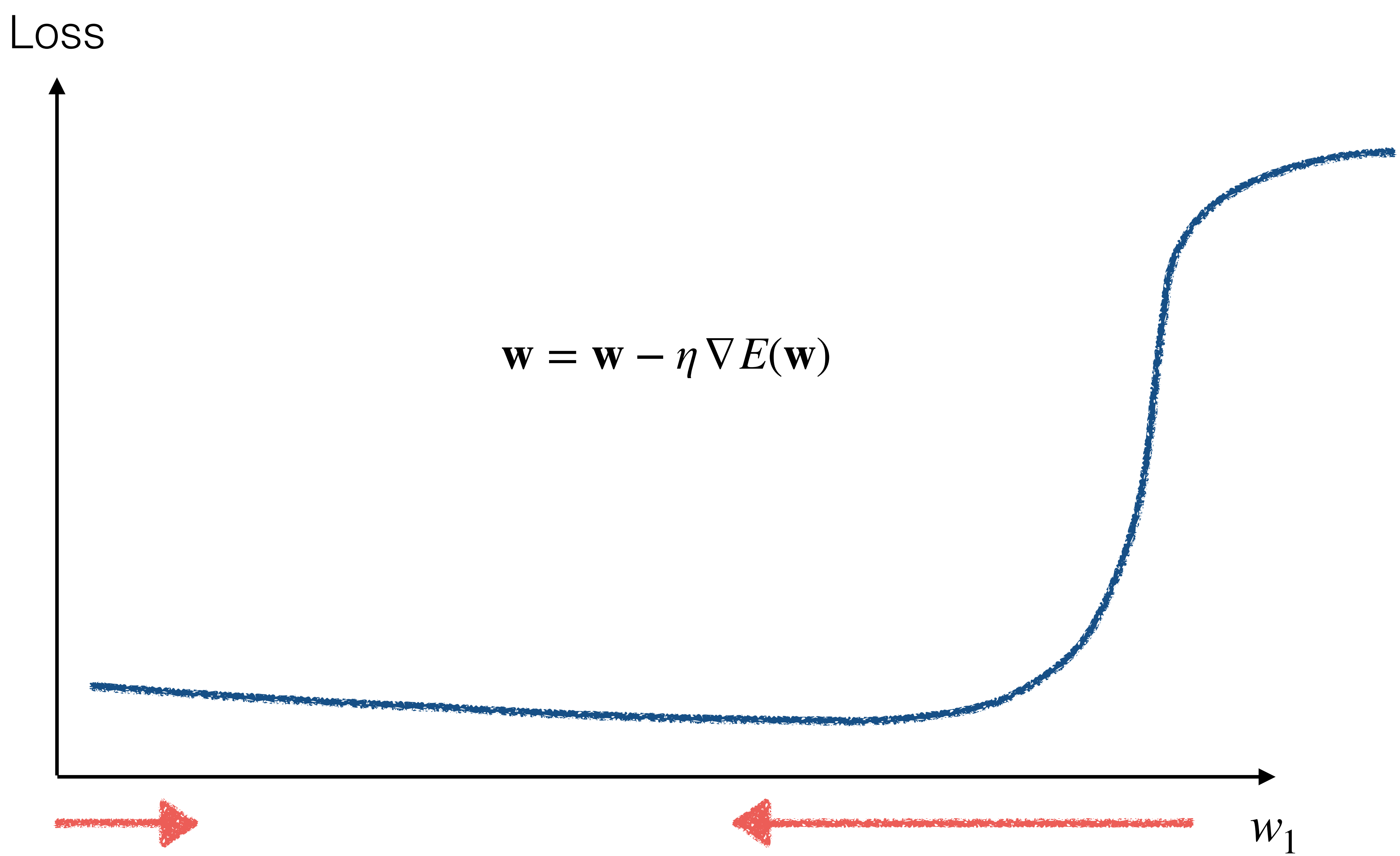
$\frac{\partial E}{\partial w_2} > \frac{\partial E}{\partial w_1}$ (depending on the location)



$$E(\mathbf{w}) = w_1^2 + 4w_2^2$$

The path of the steepest descent in most loss functions is only an instantaneous direction of best movement, and is not the best direction in the longer term!

Difficult Topologies



Smaller gradient, slow due to small steps

Big gradient, likely to overshoot

Iterative Reweighted Least Squares / Newton Raphson

- Univariate update rule:

$$w = w - \frac{E'(w)}{E''(w)}$$

Reduce/increase step size if the curvature is high/low.

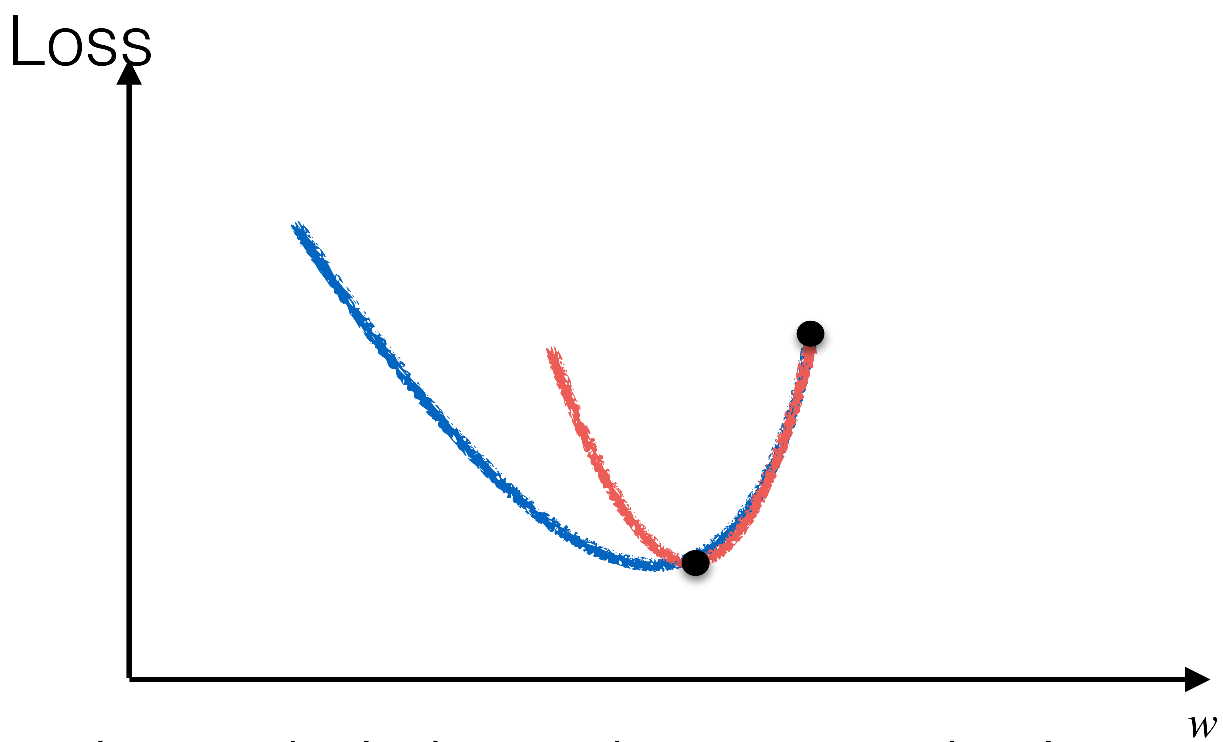
- Multivariate update rule:

$$\mathbf{w} = \mathbf{w} - H_E^{-1}(\mathbf{w}) \nabla E(\mathbf{w})$$

where $H_E^{-1}(\mathbf{w})$ is the inverse of the Hessian at the old $\mathbf{w}$ and $\nabla E(\mathbf{w})$ is the gradient at the old $\mathbf{w}$.

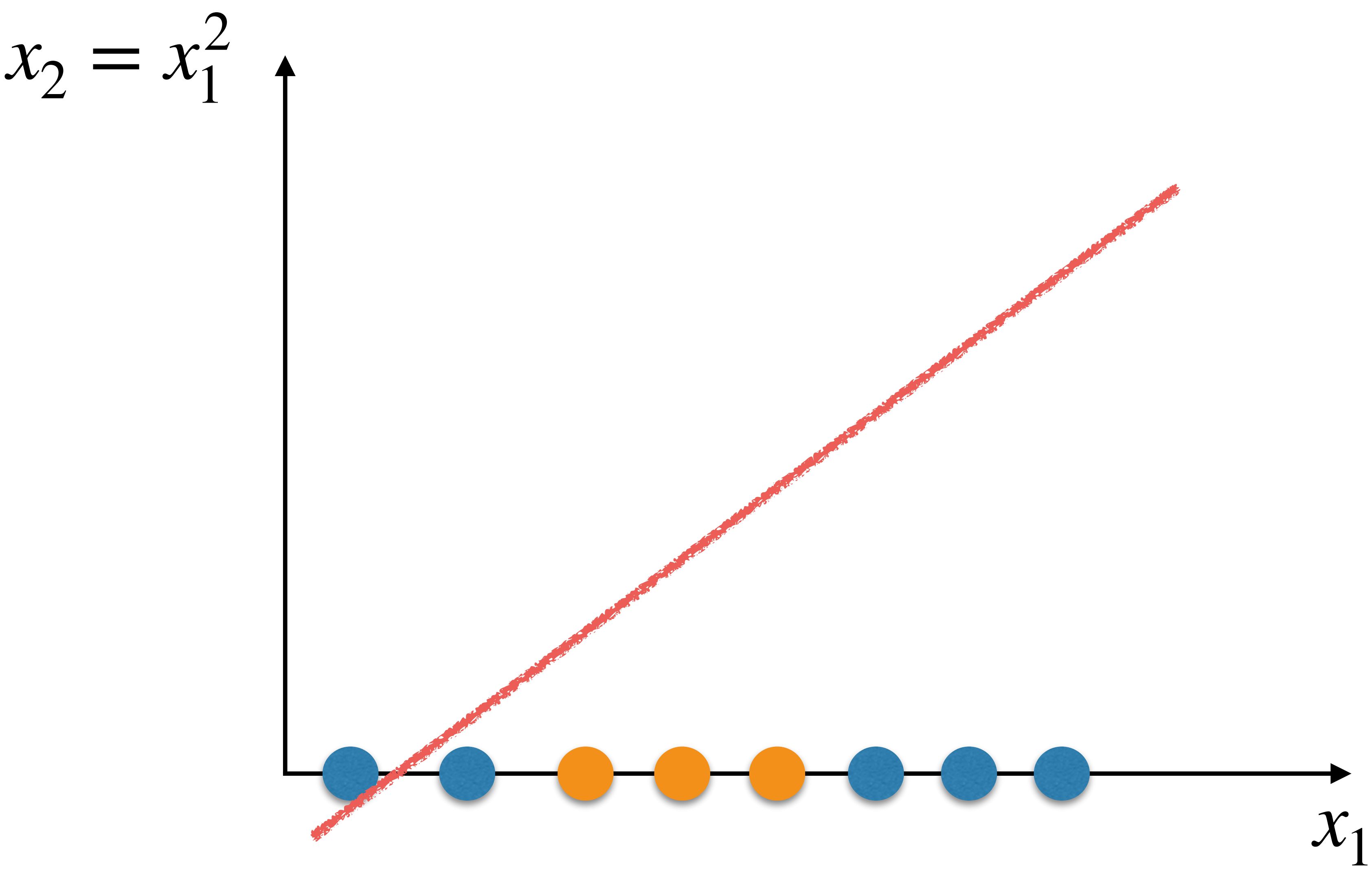
Weight Update Rule for Non-Quadratic Loss Functions

$$\mathbf{w} = \mathbf{w} - H_E^{-1}(\mathbf{w}) \nabla E(\mathbf{w})$$



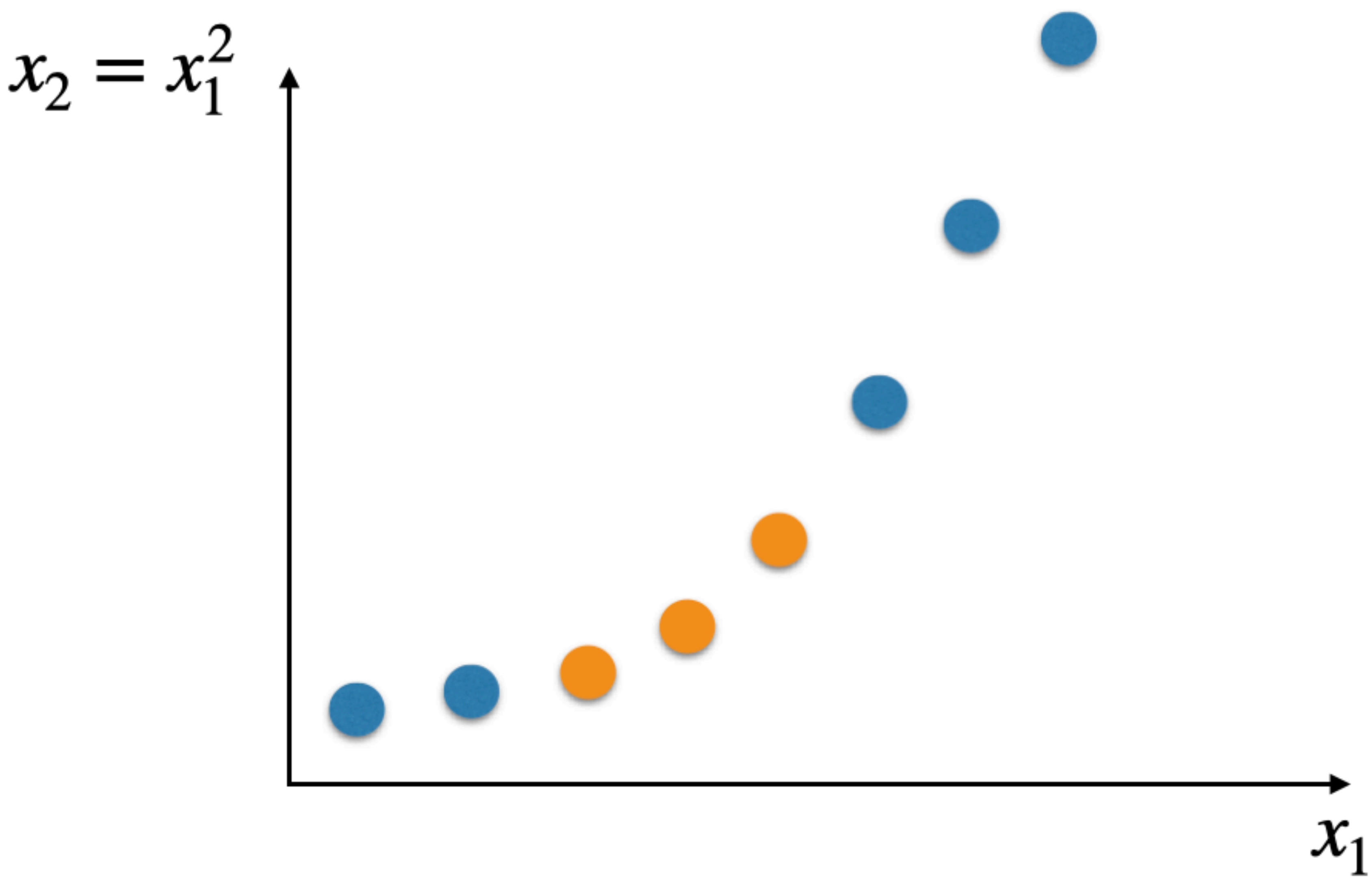
- The weight update rule is based on a quadratic approximation based on a Taylor polynomial of degree 2.
- This update will take us to the optimal of the quadratic approximation in a single step.
- However, as the quadratic approximation is not the true loss function, we will need to apply this rule iteratively.

Nonlinear Transformation / Basis Expansion



Higher dimensional embedding / feature space: $\phi(\mathbf{x}) = (x_1, x_1^2)^T$

Nonlinear Transformation / Basis Expansion

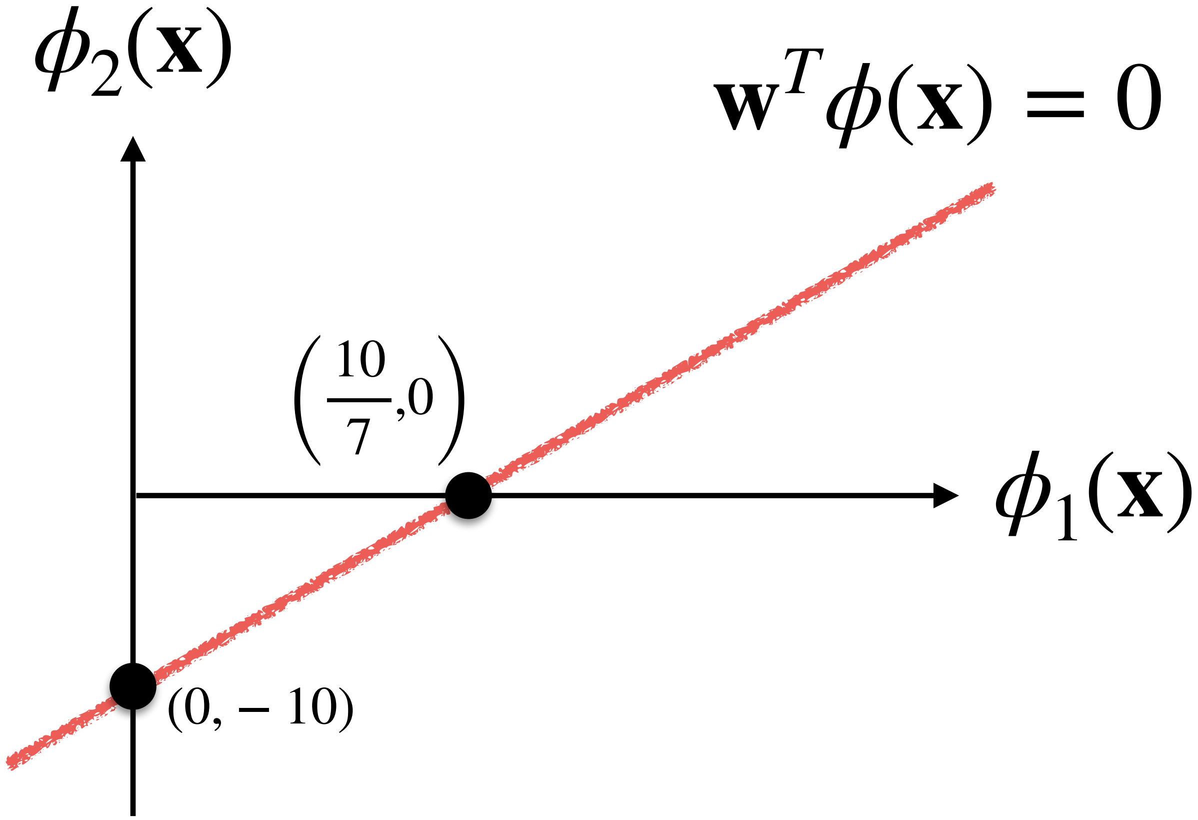


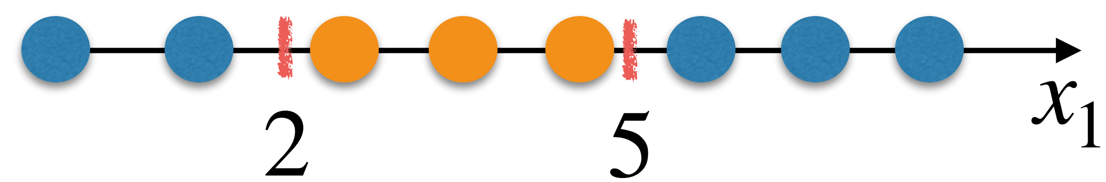
feature transform / basis expansion

basis functions

Higher dimensional embedding / feature space: $\phi(\mathbf{x}) = (x_1, x_1^2)^T$

Illustration for $\mathbf{w}^T = (10, -7, 1)$, $\phi(\mathbf{x}) = (1, x_1, x_1^2)^T$





$$w_0 \times 1 + w_1\phi_1(\mathbf{x}) + w_2\phi_2(\mathbf{x}) = 0$$

$$10 \times 1 - 7\phi_1(\mathbf{x}) + 1\phi_2(\mathbf{x}) = 0$$

$$10 - 7\phi_1(\mathbf{x}) + 1\phi_2(\mathbf{x}) = 0$$

$$w_2 x_1^2 + w_1 x_1 + w_0 \times 1 = 0$$

$$1x_1^2 - 7x_1 + 10 = 0$$

$$x_1 = \frac{7 \pm \sqrt{(-7)^2 - 4 \times 1 \times 10}}{2 \times 1}$$

Support Vector Machines

$$w_1x_1 + w_2x_2 + w_0 = 0$$ Equation of a line

$$w_1x_1 + w_2x_2 + w_3x_3 + w_0 = 0$$ Equation of a plane

$$w_1x_1 + w_2x_2 + \cdots + w_dx_d + w_0 = 0$$ Equation of a hyperplane

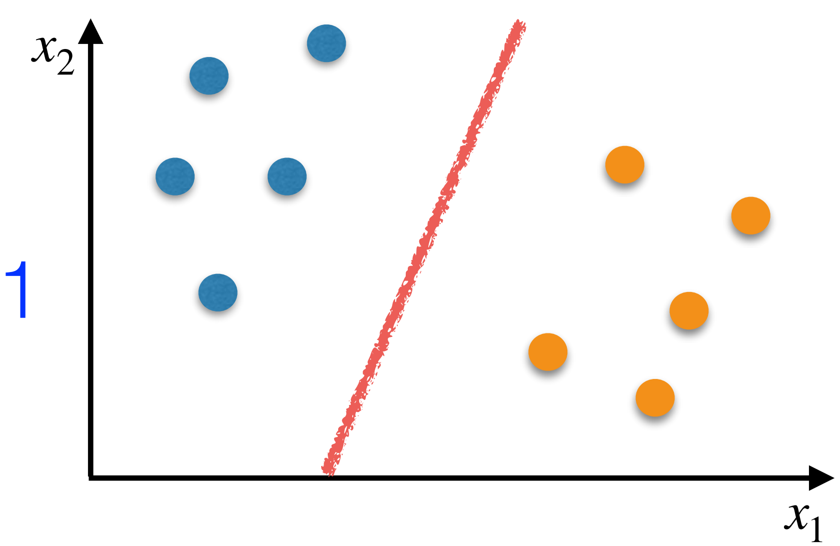
$$\mathbf{w}^T\mathbf{x} + w_0$$

$$\mathbf{w}^T\mathbf{x} + b$$

bias

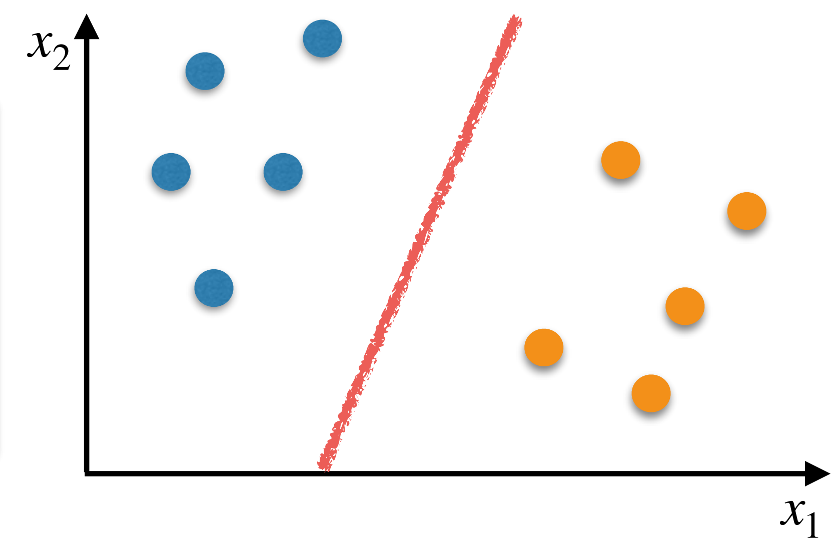
$\mathbf{w}^T\mathbf{x} + b > 0 \rightarrow$ class +1

$\mathbf{w}^T\mathbf{x} + b < 0 \rightarrow$ class -1



Hypothesis Set

$$h(\mathbf{x}) = \begin{cases} +1 & \text{if } \mathbf{w}^T\mathbf{x} + b > 0 \\ -1 & \text{if } \mathbf{w}^T\mathbf{x} + b < 0 \end{cases}, \quad \forall \mathbf{w} \in \mathbb{R}^d, \forall b \in \mathbb{R}$$

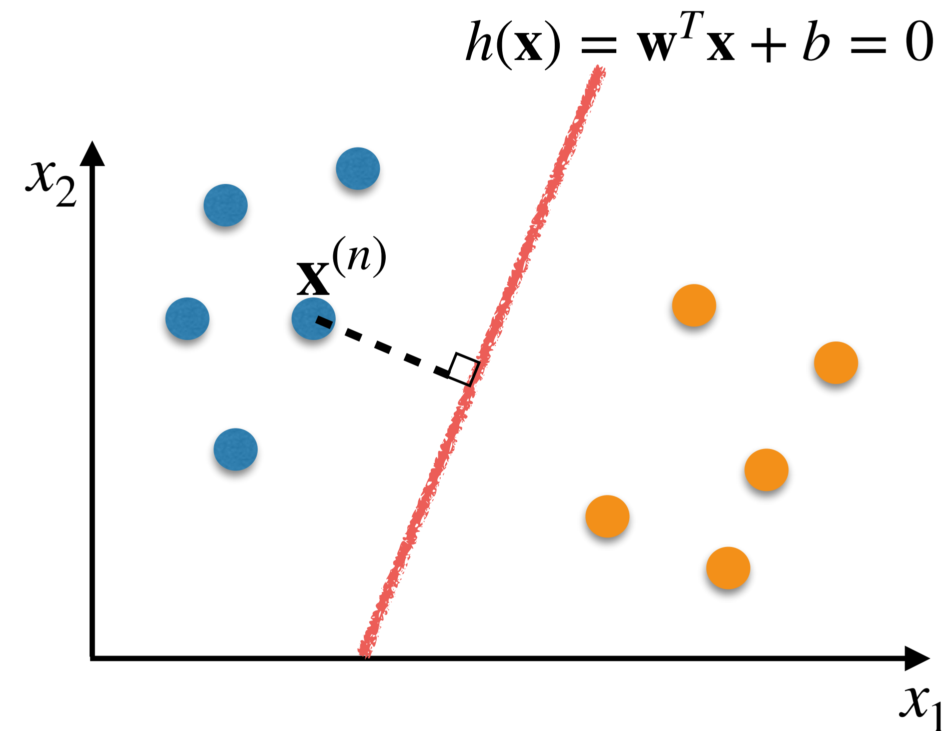


For simplicity, we will use the notation $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$.

Perpendicular Distance From a Point $\mathbf{x}^{(n)}$ to a Hyperplane $h(\mathbf{x}) = 0$

$$\text{dist}(h, \mathbf{x}^{(n)}) = \frac{|\, h(\mathbf{x}^{(n)}) \,|}{||\mathbf{w}||}$$

where $||\mathbf{w}|| = \sqrt{\mathbf{w}^T \mathbf{w}}$ is the Euclidean norm (the length of the vector $\mathbf{w}$)



Find $\mathbf{w}$ and b that maximise the margin, i.e., the perpendicular distance between the hyperplane and the closest training example.

Constraint: all training examples must be correctly classified.

Subject to $y^{(n)} h(\mathbf{x}^{(n)}) > 0$, $\forall (\mathbf{x}^{(n)}, y^{(n)}) \in \mathcal{T}$.

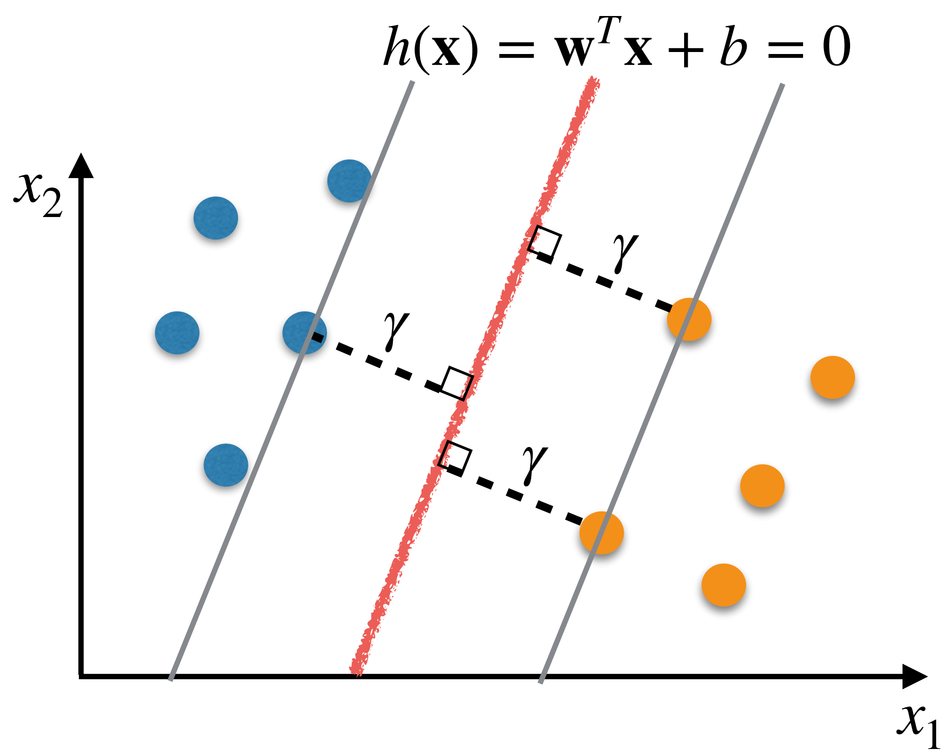
$$y^{(n)} = +1 \;\; h(\mathbf{x}^{(n)}) > 0$$

Maximum Margin Classifier

$$\underset{\mathbf{w},b}{\text{argmin}} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 \right\}$$

Subject to $y^{(n)} h(\mathbf{x}^{(n)}) \geq 1$

for $\forall (\mathbf{x}^{(n)}, y^{(n)}) \in \mathcal{T}$.



The Dual Representation

Lagrangian function

Kernel function

$$\underset{\mathbf{a}}{\operatorname{argmax}}\, \tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a^{(n)} - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a^{(n)}a^{(m)}y^{(n)}y^{(m)}k(\mathbf{x}^{(n)}, \mathbf{x}^{(m)})$$

Inner product

Where: $k(\mathbf{x}^{(n)}, \mathbf{x}^{(m)}) = \phi(\mathbf{x}^{(n)})^T\phi(\mathbf{x}^{(m)})$

Subject to: $a^{(n)} \geq 0, \forall n \in \{1, \cdots, N\}$ $\quad \sum_{n=1}^{N} a^{(n)}y^{(n)} = 0$

There is a way to compute $k(\mathbf{x}^{(n)}, \mathbf{x}^{(m)}) = \phi(\mathbf{x}^{(n)})^T\phi(\mathbf{x}^{(m)})$ without having to ever compute $\phi(\mathbf{x})$. This is called the Kernel Trick.

Mercer's Condition

- Consider any finite set of points $\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(M)}$ (not necessarily the training set).
- Gram matrix: An $M \times M$ similarity matrix $\mathbf{K}$, whose elements are given by $K_{i,j} = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$.
- Mercer's condition states that $\mathbf{K}$ must be symmetric and positive semidefinite.
  - Symmetric: $k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = k(\mathbf{x}^{(j)}, \mathbf{x}^{(i)})$.
  - Positive semidefinite: $\mathbf{z}^T\mathbf{K}\mathbf{z} \geq 0, \forall \mathbf{z} \in \mathbb{R}^M$.

If these conditions are satisfied, the inner product defined by the kernel in the feature space respects the properties of inner products.

Kernels as Similarity Functions

- Gaussian kernel, a.k.a. Radial Basis Function (RBF) kernel.

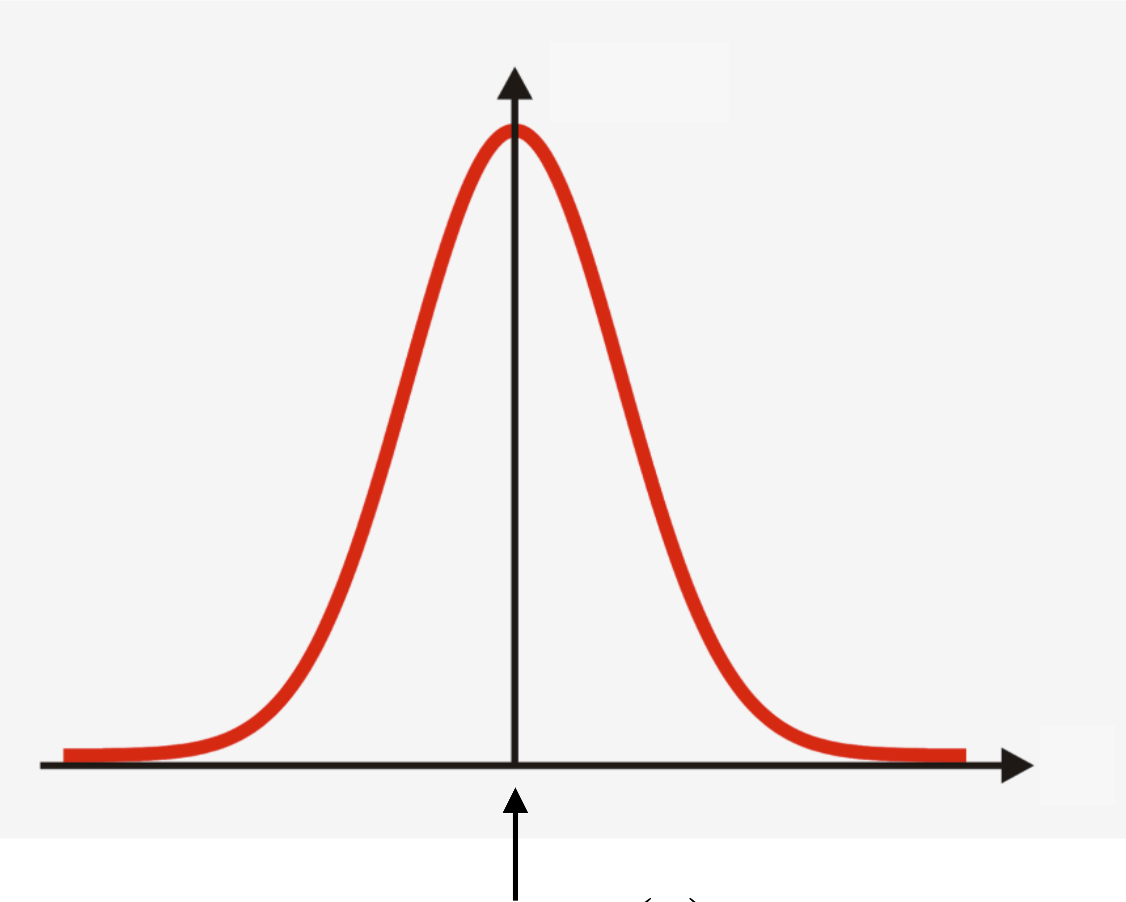
$$k(\mathbf{x}, \mathbf{x}^{(n)}) = e^{-\frac{\|\mathbf{x} - \mathbf{x}^{(n)}\|^2}{2\sigma^2}}$$

The embedding ϕ is infinite dimensional!

Taylor series with infinite terms gives a representation of the true Gaussian itself.

E.g., for $\sigma = 1$

$$k(\mathbf{x}, \mathbf{x}^{(n)}) = \sum_{j=0}^{\infty} \frac{(\mathbf{x}^T \mathbf{x}^{(n)})^j}{j!} e^{-\frac{1}{2}\|\mathbf{x}\|^2} e^{-\frac{1}{2}\|\mathbf{x}^{(n)}\|^2}$$



$x = x^{(n)}$

Kernels as Similarity Functions

- Kernels for sets
- Kernels for text
- Kernels for strings
- Kernels for trees
- Kernels for graphs

Making Predictions

$$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

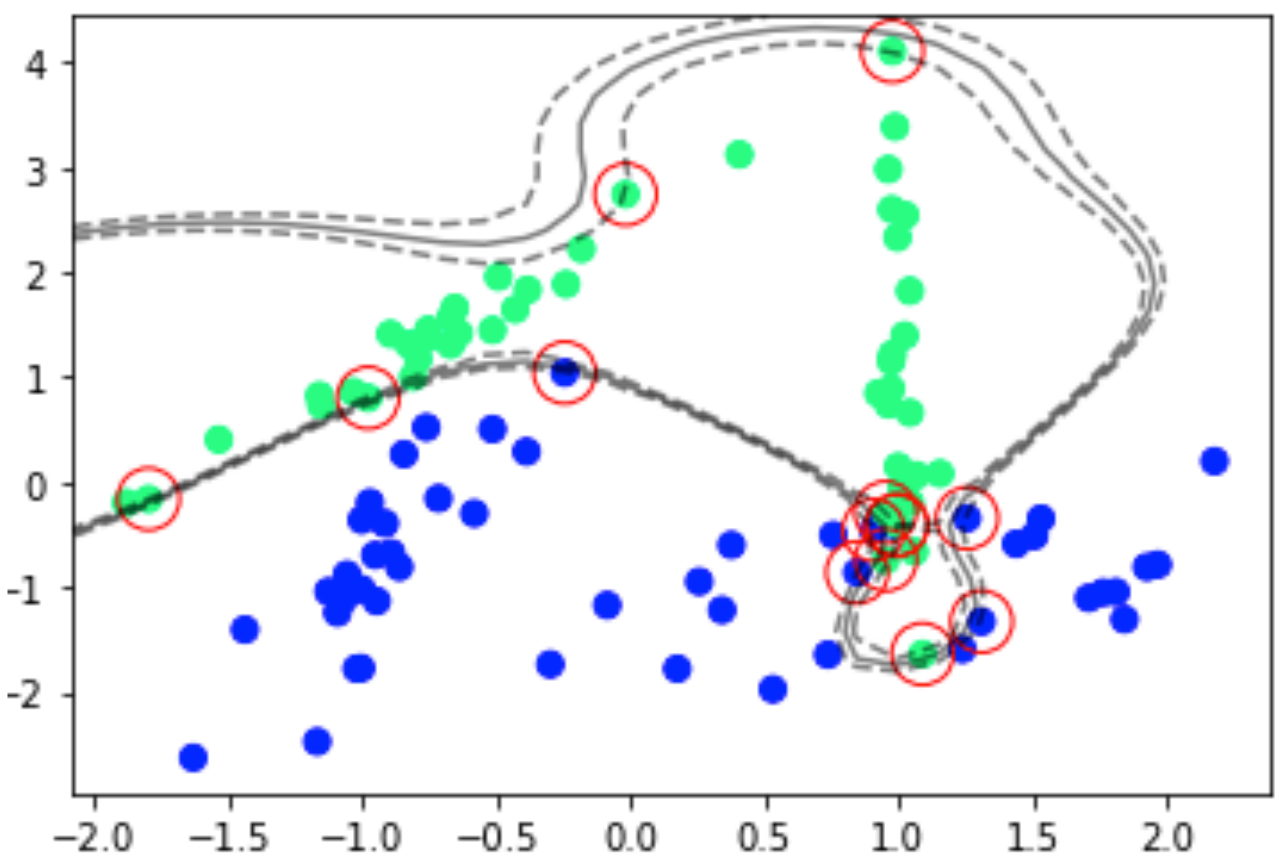
$\mathbf{w}^T\mathbf{x} + b > 0 \rightarrow$ class +1

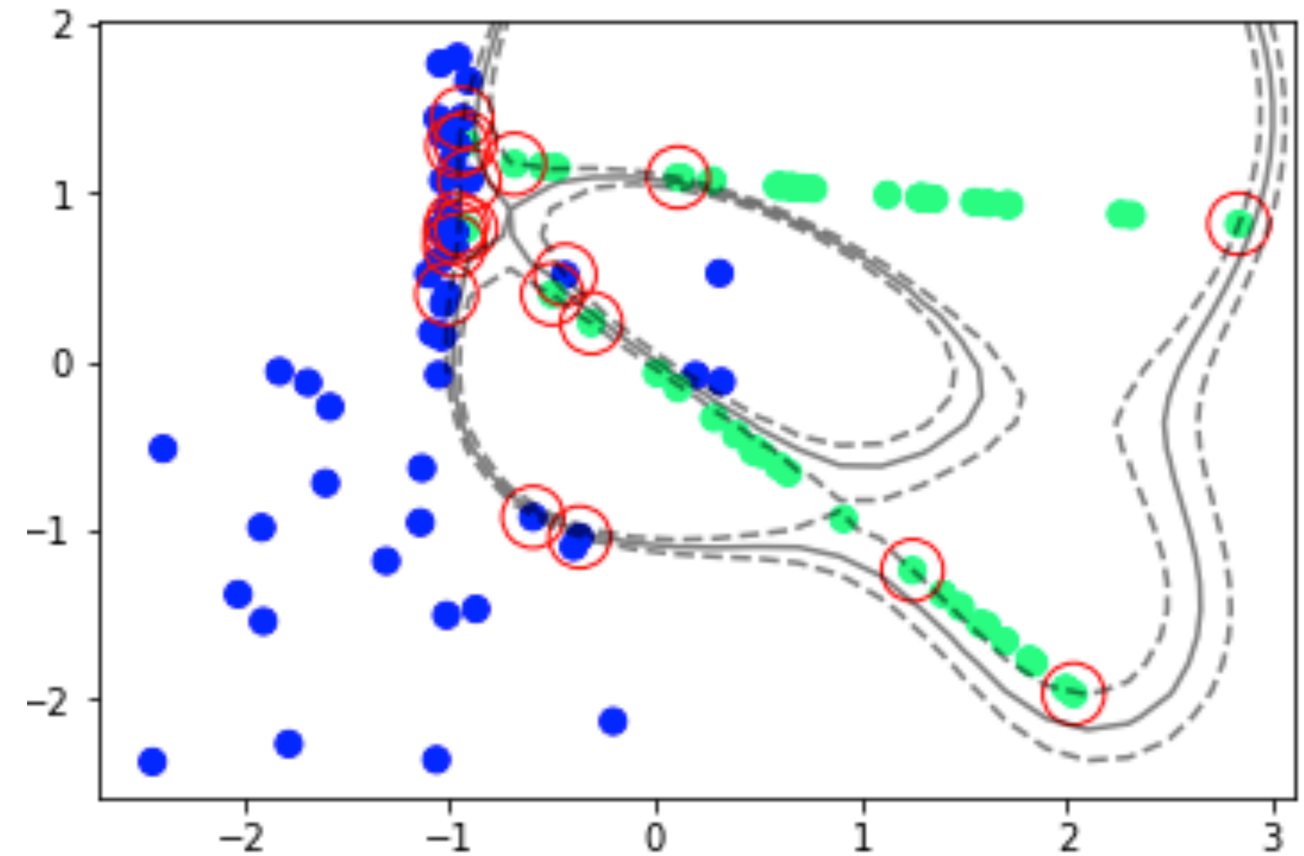
$\mathbf{w}^T\mathbf{x} + b < 0 \rightarrow$ class -1

$$h(\mathbf{x}) = \sum_{n \in S} a^{(n)} y^{(n)} k(\mathbf{x}, \mathbf{x}^{(n)}) + b$$

Overfitting

Using Gaussian kernel: $k(\mathbf{x}^{(n)}, \mathbf{x}^{(m)}) = e^{-\frac{\|\mathbf{x}^{(n)} - \mathbf{x}^{(m)}\|^2}{2\sigma^2}}$





Code adapted from: https://jakevdp.github.io/PythonDataScienceHandbook/05.07-support-vector-machines.html

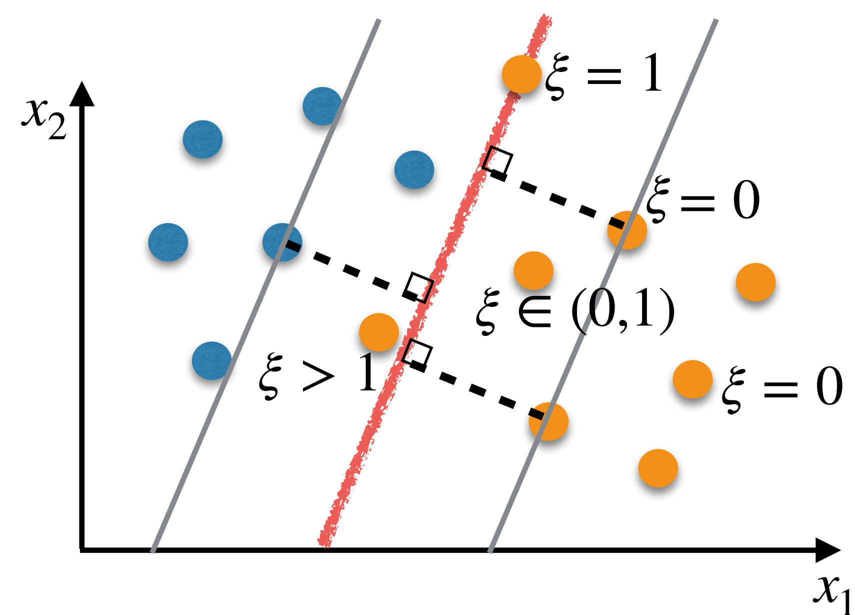
* Red circles represent the support vectors

Soft Margin SVM

- Recap of our optimisation problem (primal representation):

$$\underset{\mathbf{w},b}{\text{argmin}} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 \right\}$$

Subject to: $y^{(n)} h(\mathbf{x}^{(n)}) \geq 1,\ \forall n \in \{1,2,\cdots,N\}$



- Using Slack

$$\underset{\mathbf{w},b,\xi}{\text{argmin}} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^{N} \xi^{(n)} \right\}$$

Subject to: $y^{(n)} h(\mathbf{x}^{(n)}) \geq 1 - \xi^{(n)},\ \forall n \in \{1,2,\cdots,N\}$

$$\xi^{(n)} \geq 0$$

C is a hyperparameter that controls the trade-off between the slack variable penalty and the margin

Overview of the Module

Definition and components of supervised learning

Classification approaches and underlying optimisation algorithms

Regression approaches and underlying optimisation algorithms

Foundational theory