

Neural Networks and Learning Systems
TBM126 / 732A55
2024

Lecture 2
Supervised learning –
Linear classifiers

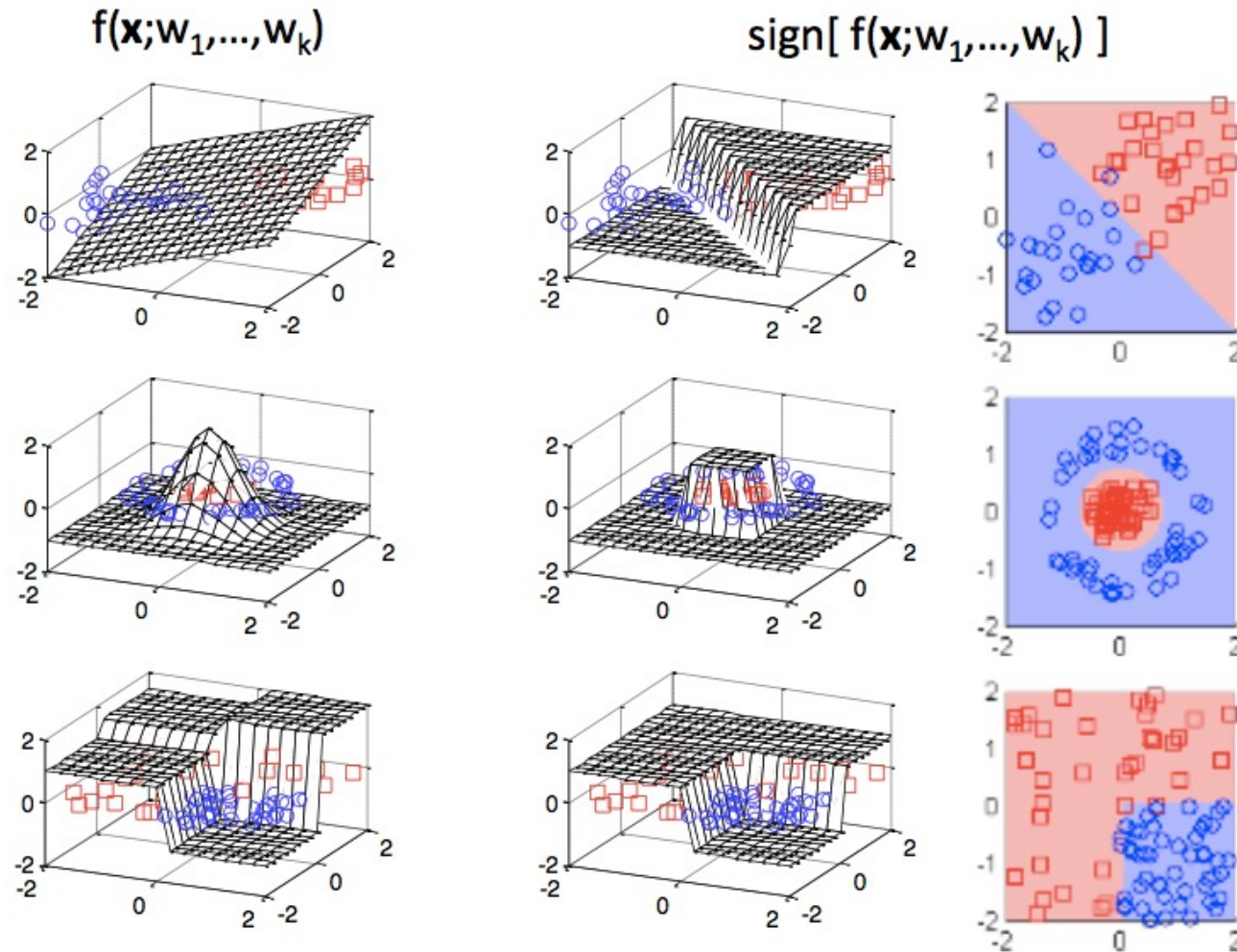
Magnus Borga
magnus.borga@liu.se

Recap - Supervised learning

- **Task:** Learn to predict/classify new data from labelled examples.
- **Input:** Training data examples $\{\mathbf{x}_i, y_i\} \ i=1...K$, where \mathbf{x}_i is a feature vector and y_i is a class label in the set Ω . Today we assume two classes: $\Omega = \{-1, 1\}$
- **Output:** A discriminant function $\text{sign}[f(\mathbf{x}; w_1, \dots, w_N)] \rightarrow \Omega$

Find a function f and adjust the parameters w_1, \dots, w_N so that new feature vectors are classified correctly. Generalization!

The function $f(\mathbf{x}; w_1, \dots, w_k)$



Linear classifier

Advantages of a parametric function

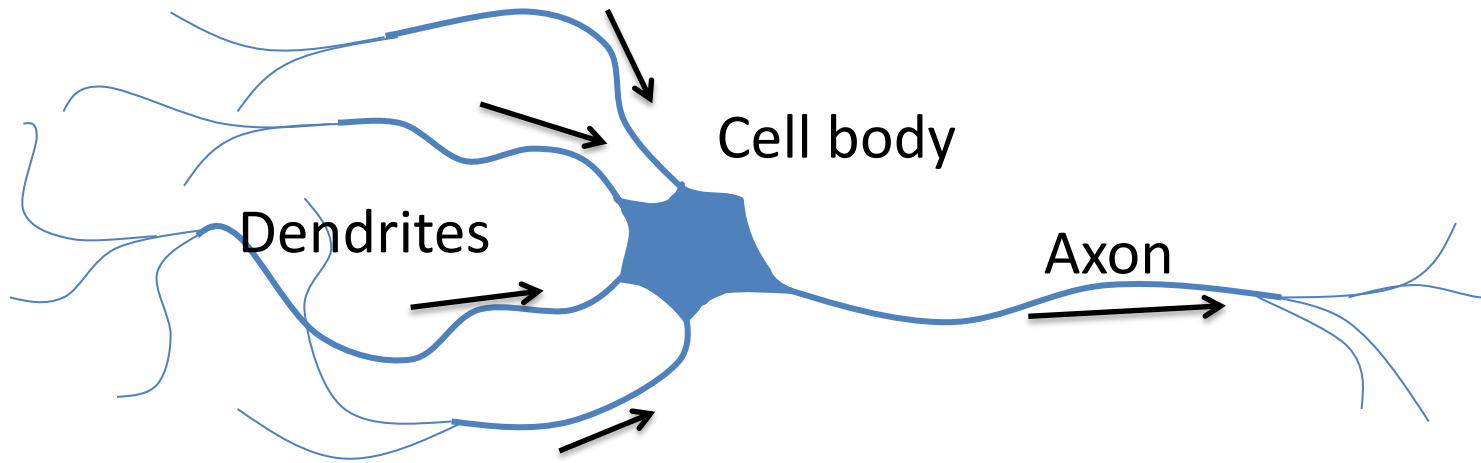
$$f(\mathbf{x}; w_1, \dots, w_N)$$

- Only stores a few parameters (w_0, w_1, \dots, w_N) instead of all the training samples, as in k -NN.
- Easy to evaluate on which side of the line a new sample is on, for example $\mathbf{w}^T \mathbf{x} < 0$ or $\mathbf{w}^T \mathbf{x} > 0$ for a linear function.

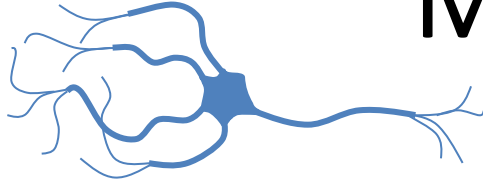
How does the brain take decisions?

(on the low level!)

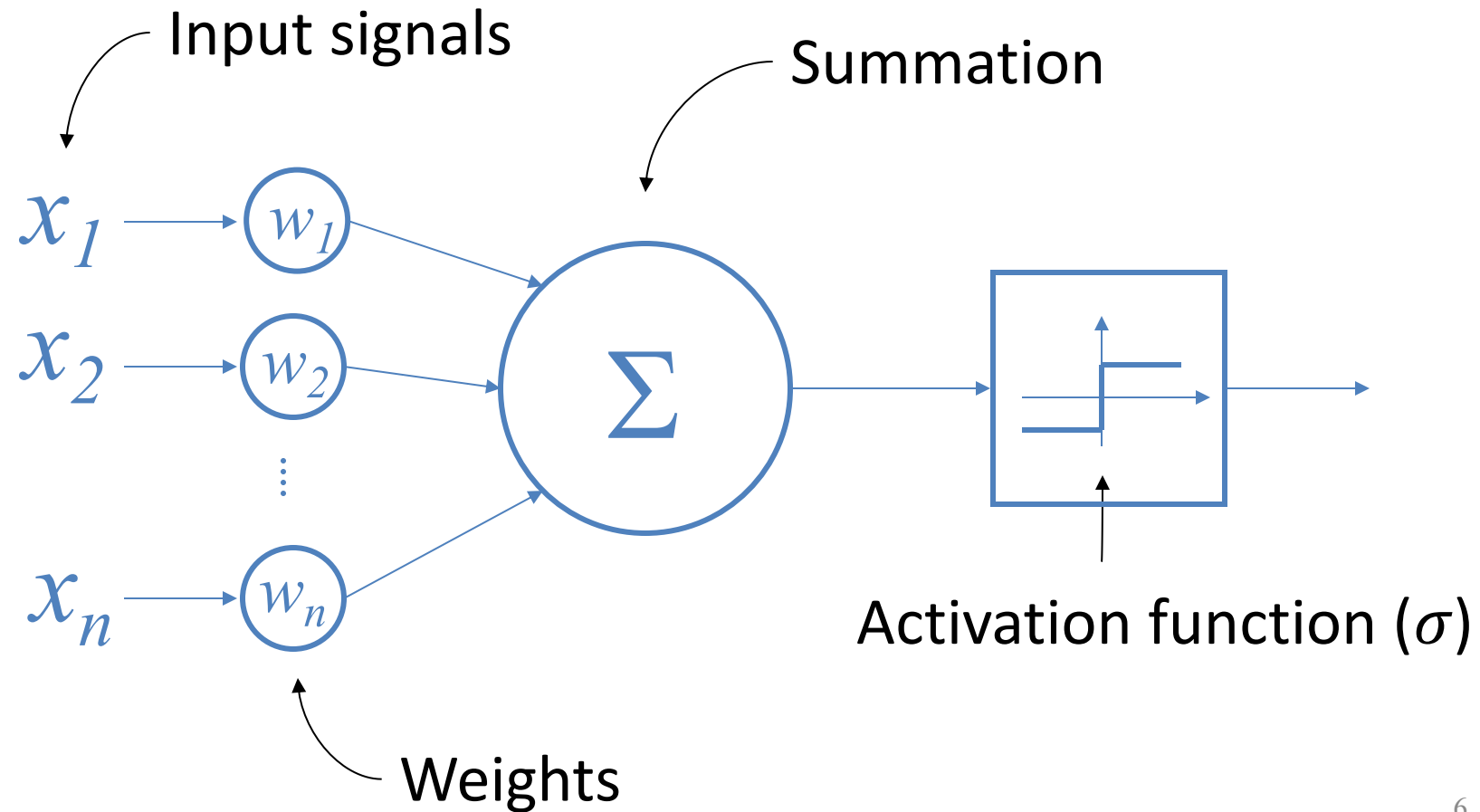
- Basic unit: the neuron



- The human brain has approximately 100 billion (10^{11}) neurons.
- Each neuron connected to about 7000 other neurons.
- Approx. 10^{14} - 10^{15} synapses (connections).



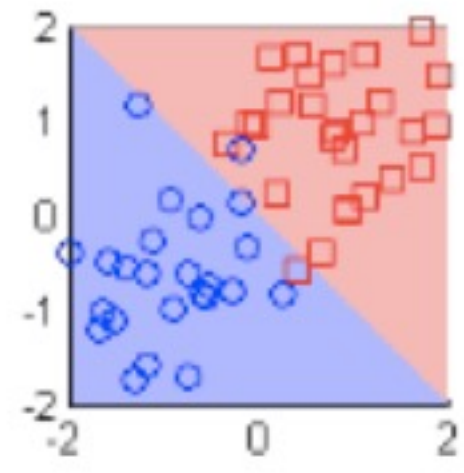
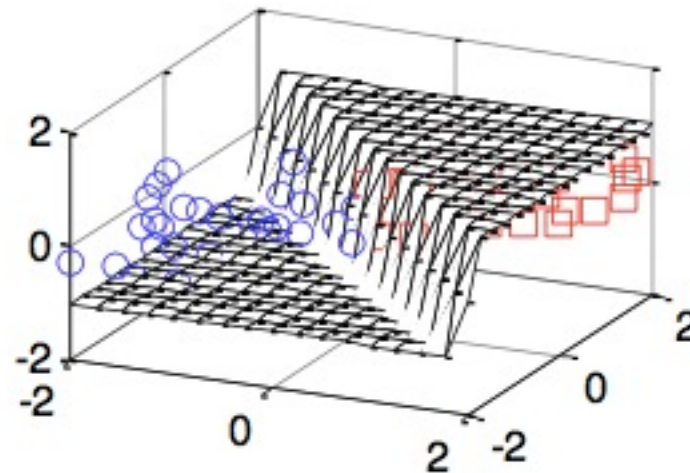
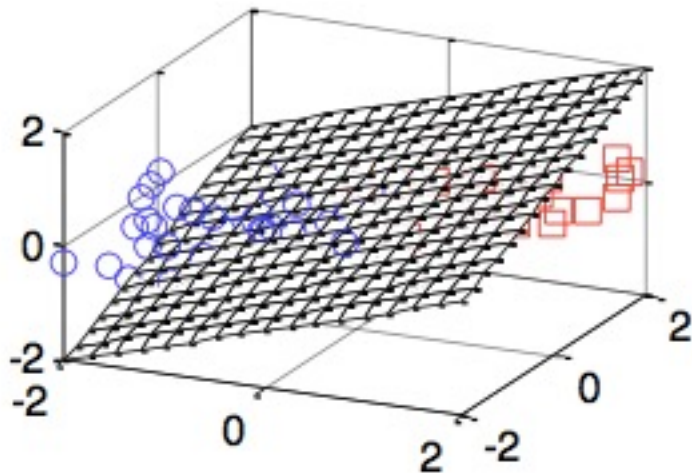
Model of a neuron



The Perceptron

(McCulloch & Pitts 1943, Rosenblatt 1962)

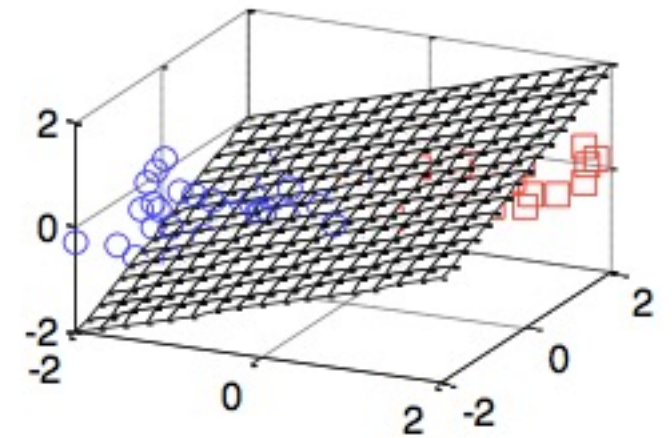
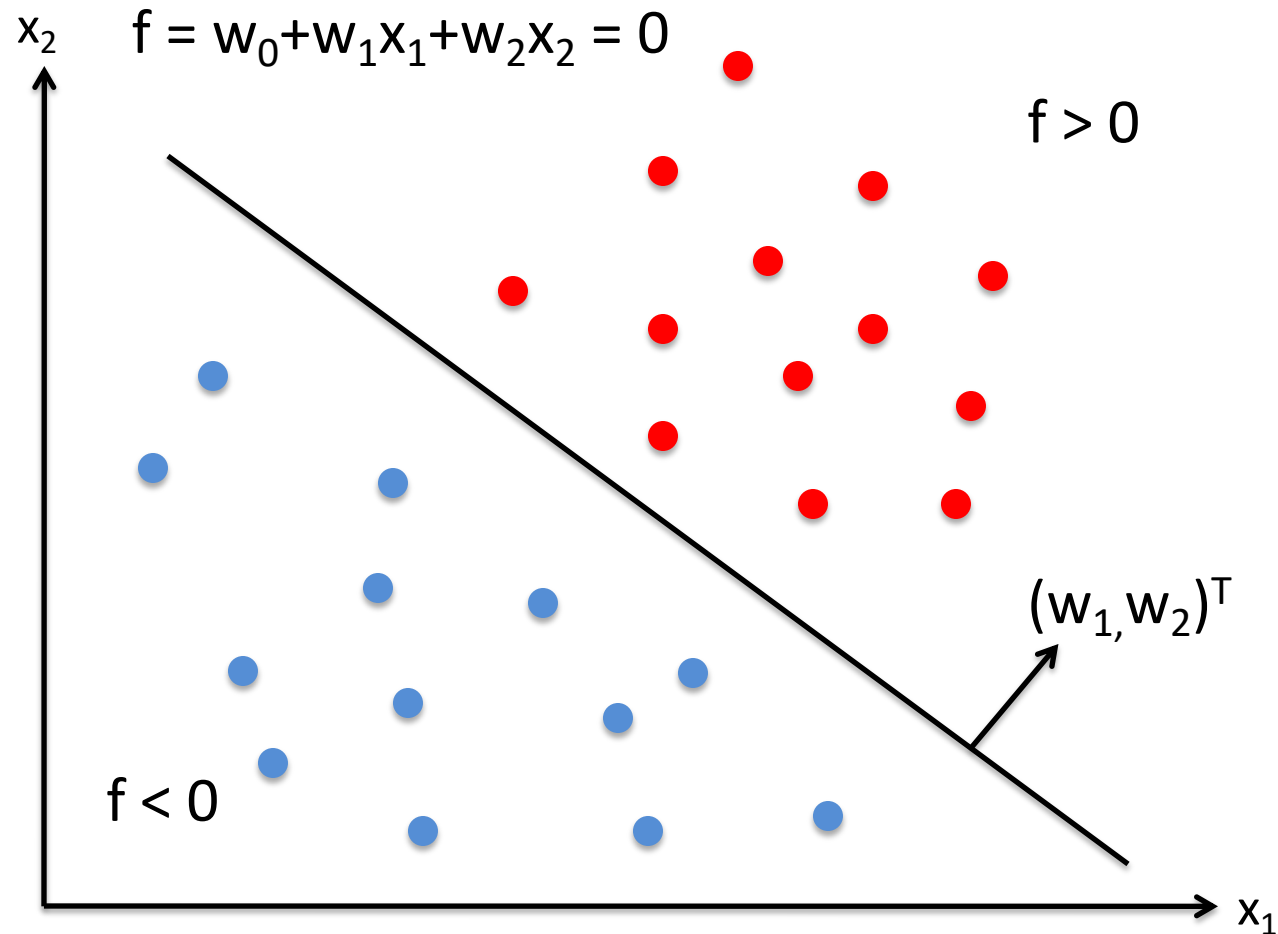
$$f(x_1, \dots, x_n; w_0, \dots, w_n) = \sigma \left(w_0 + \sum_{i=1}^n w_i x_i \right) = \sigma(w_0 + \mathbf{w}^T \mathbf{x})$$



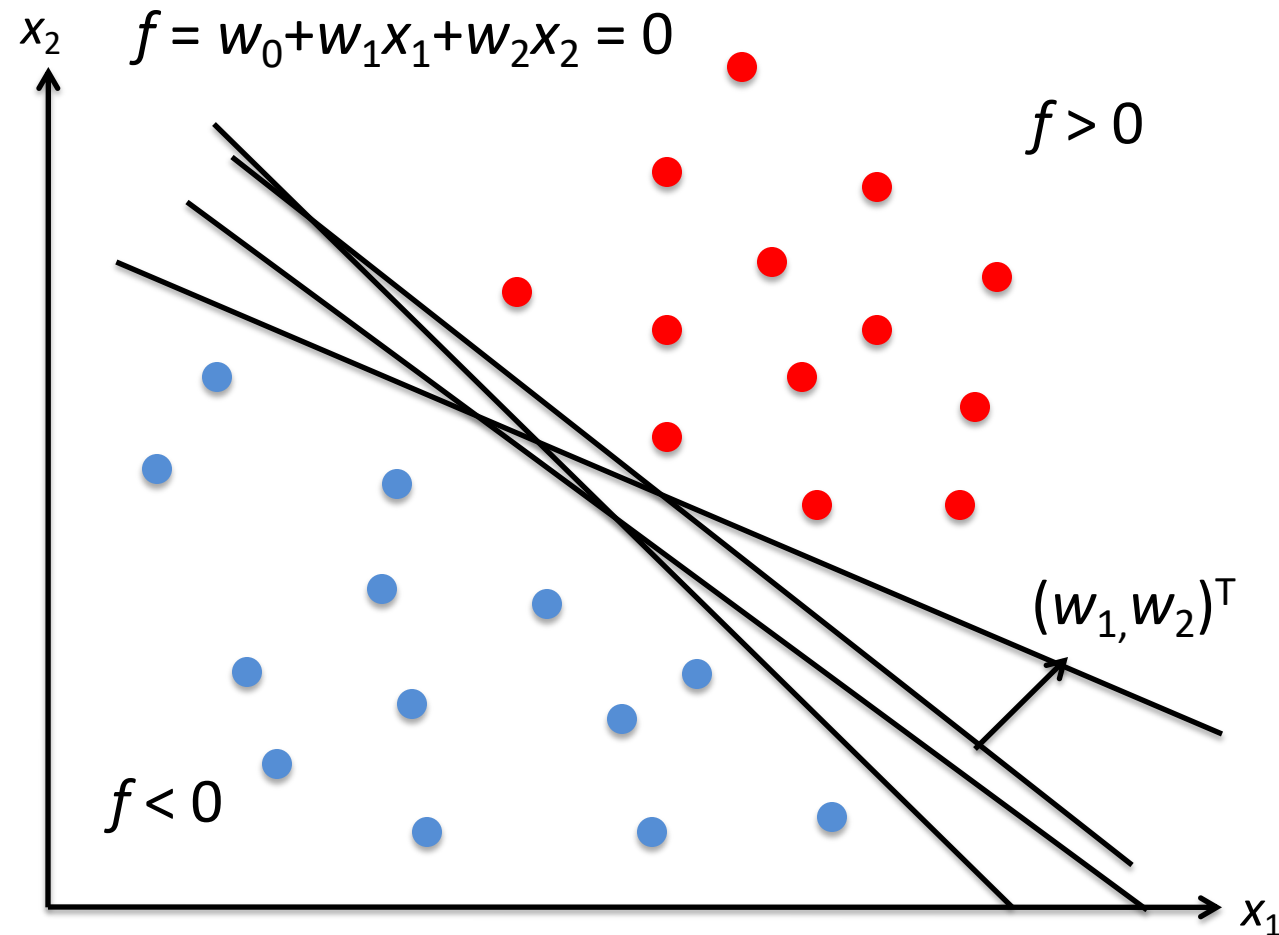
Extra reading on the history of the perceptron:

<http://www.csulb.edu/~cwallis/artificialn/History.htm>

Geometry of linear classifiers

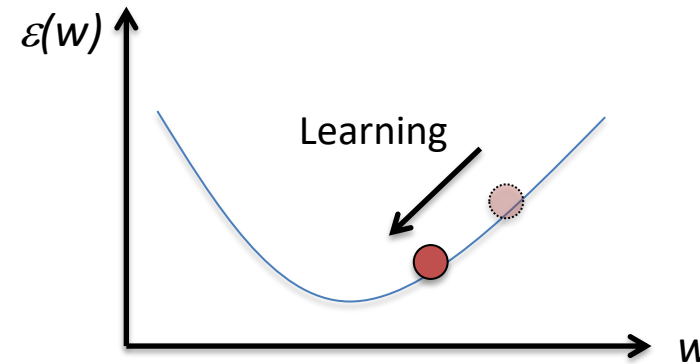


Which linear classifier to choose?



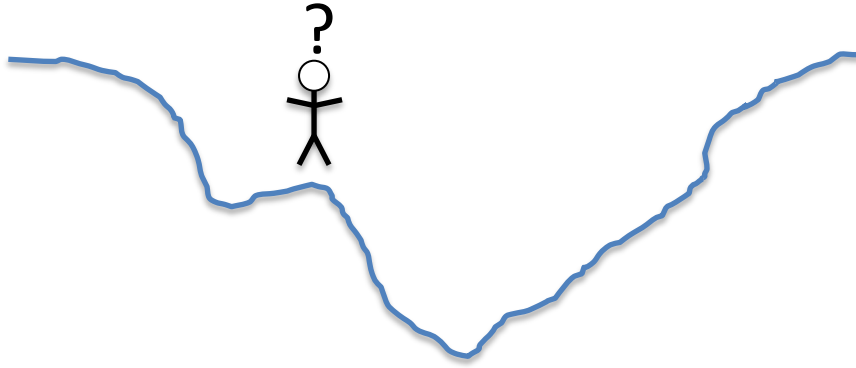
Find the best separator – optimization!

- Min of a loss function $\varepsilon(w_0, w_1, \dots, w_n)$ with the weights w_0, w_1, \dots, w_n as parameters.

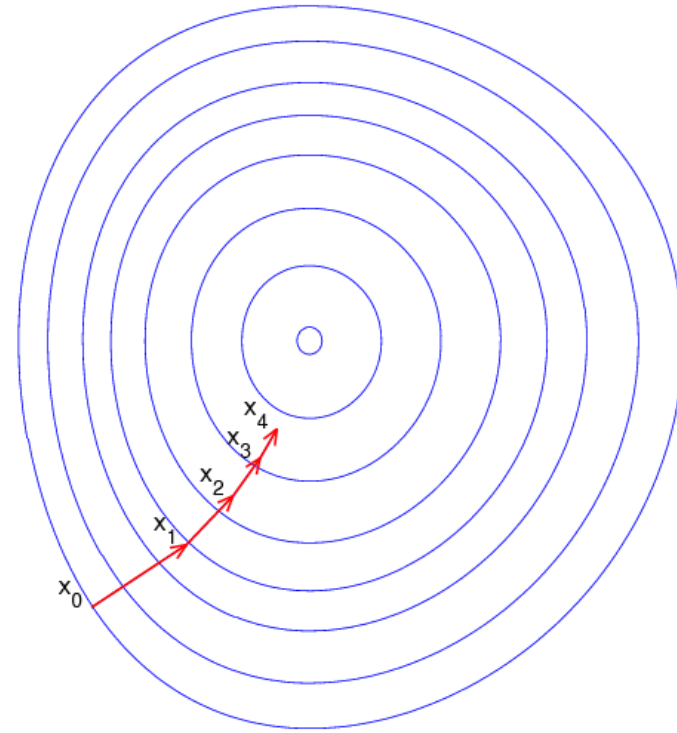


- Ways to optimize:
 - Algebraic: Set derivative $\frac{\partial \varepsilon}{\partial w_i} = 0$ and solve.
 - Brute force: Try many values systematically and choose the best.
 - Iterative: Follow the gradient direction until the minimum of ε is reached.

Gradient descent



How to get to the lowest point?



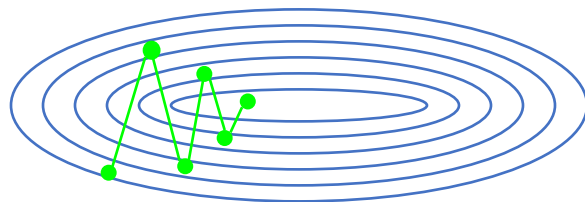
$$\nabla \mathcal{E} = \frac{\partial \mathcal{E}}{\partial \mathbf{w}} = \begin{pmatrix} \frac{\partial \mathcal{E}}{\partial w_1} \\ \frac{\partial \mathcal{E}}{\partial w_2} \end{pmatrix}$$

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \eta \frac{\partial \mathcal{E}}{\partial \mathbf{w}}$$

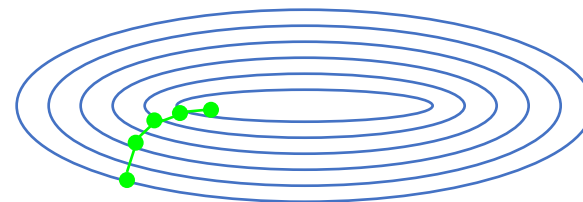
Gradient descent

Choosing the step length

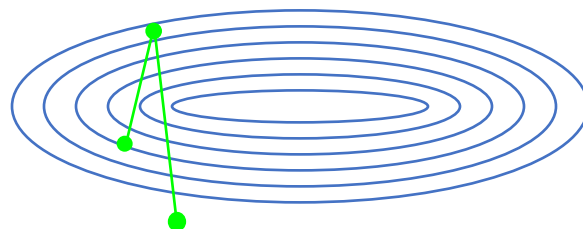
Large η



Small η

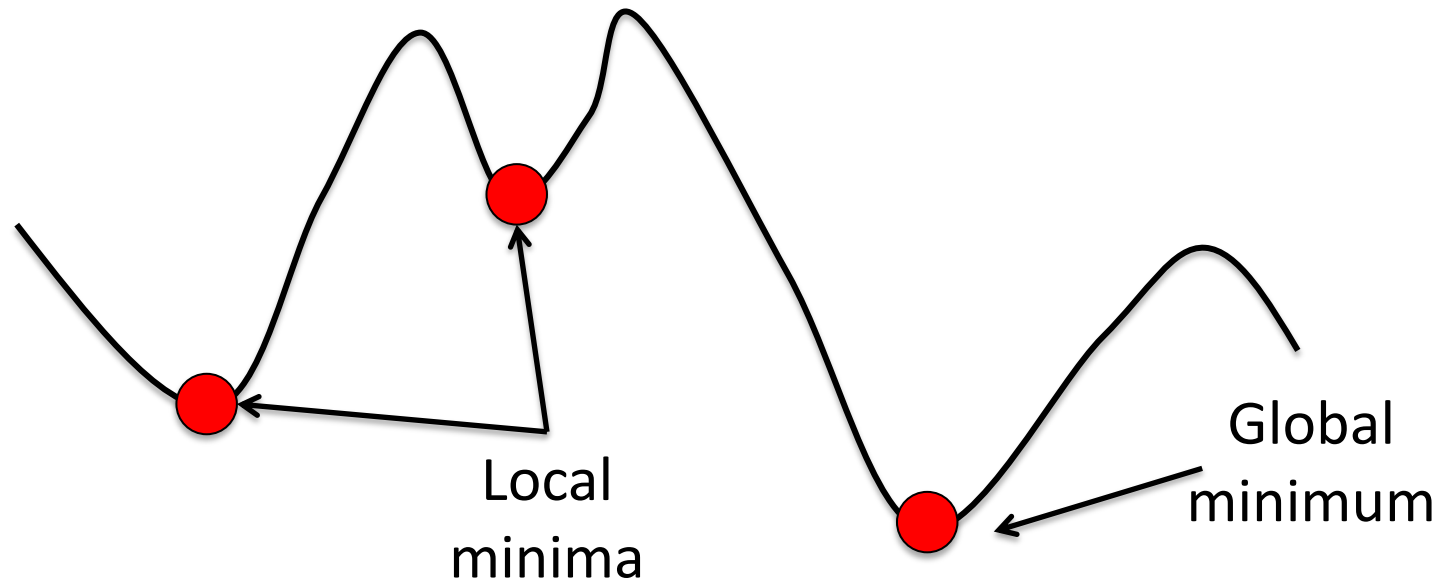


Too large η



Local optima

- Gradient descent is not guaranteed to find the global minimum.
- With a sufficiently small step length, the closest local minimum will be found.



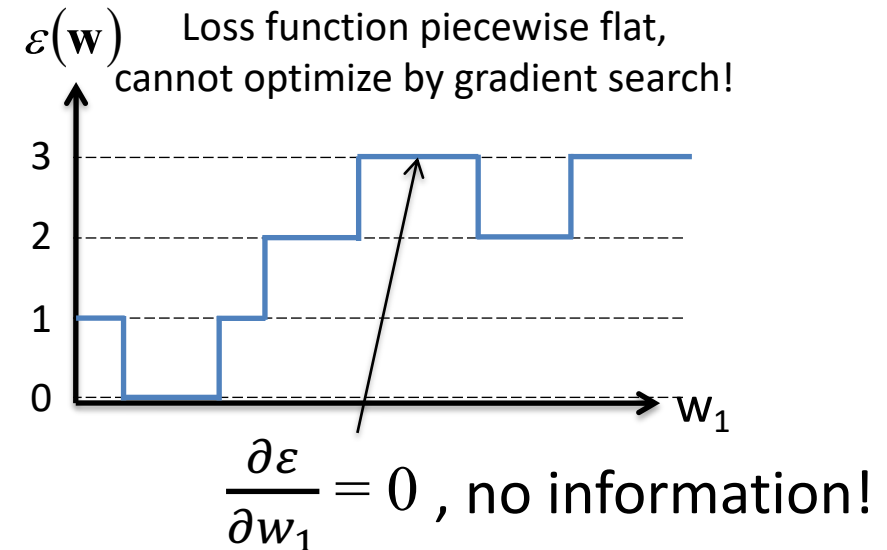
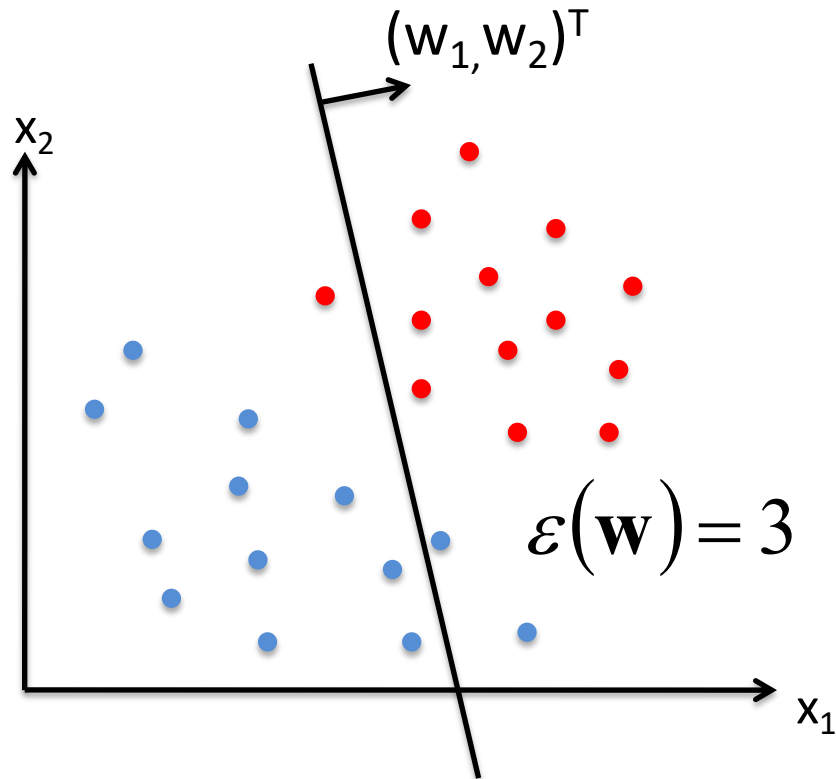
Many different loss functions $\varepsilon(\mathbf{w})$

- 0-1 loss function / empirical risk
- Mean square error
- Maximum margin
- ...

0-1 loss function / empirical risk

$\varepsilon(\mathbf{w}) = \sum_{i=1}^K I(f(\mathbf{x}_i; \mathbf{w}) \neq y_i)$ = Number of wrong classifications

Note: K is labeled "empirical" because it is over the observed training data. $I(\cdot)$ is labeled "0/1 function".



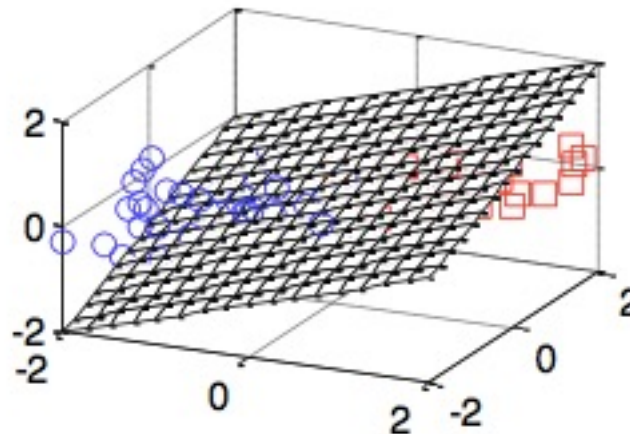
Square error loss

Minimize the following loss function

$$\mathcal{E}(\mathbf{w}) = \sum_{i=1}^K \left(\mathbf{w}^T \mathbf{x}_i - y_i \right)^2$$

$K = \#$ training samples

$y_i \in \{-1, 1\}$ depending on the class of training sample i



Minimization algorithm

$$\mathcal{E}(\mathbf{w}) = \sum_{i=1}^K \left(\mathbf{w}^T \mathbf{x}_i - y_i \right)^2$$

$$\frac{\partial \mathcal{E}}{\partial \mathbf{w}} = 2 \sum_{i=1}^K \left(\mathbf{w}^T \mathbf{x}_i - y_i \right) \mathbf{x}_i \quad \longleftarrow \text{Exercise!}$$

Gradient descent:

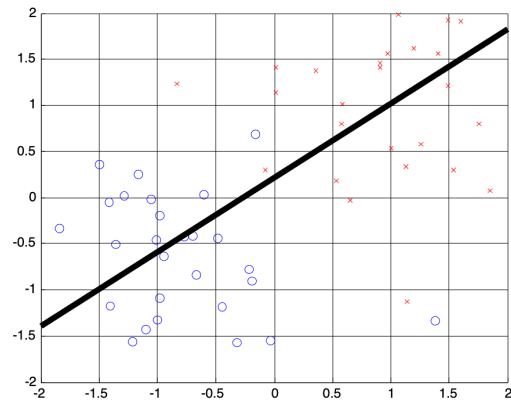
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \frac{\partial \mathcal{E}}{\partial \mathbf{w}} = \mathbf{w}_t - \eta \sum_{i=1}^K \left(\mathbf{w}_t^T \mathbf{x}_i - y_i \right) \mathbf{x}_i \quad (\text{Eq.1})$$

Algorithm:

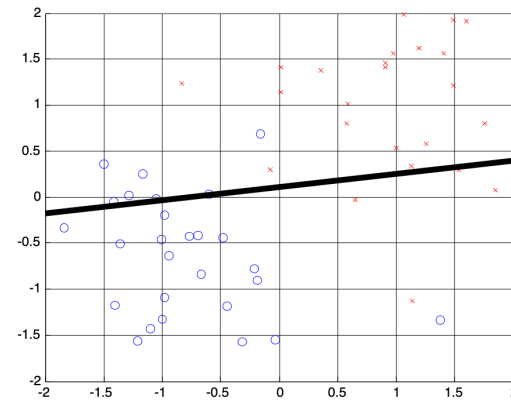
1. Start with a random \mathbf{w}
2. Iterate Eq. 1 until convergence

Example

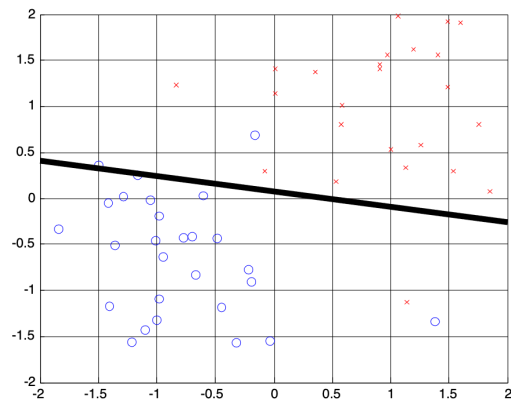
Random init



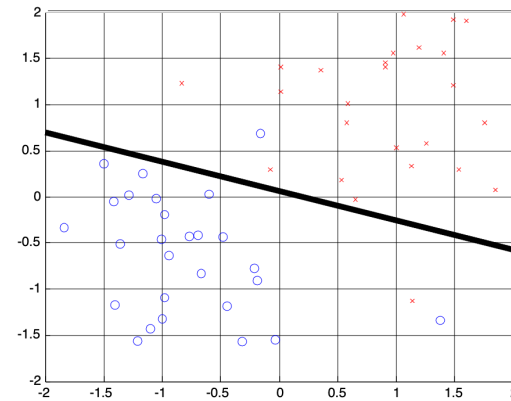
First iteration



Second iteration

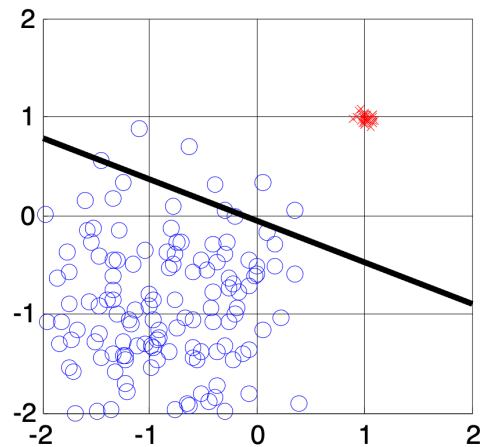


Third iteration

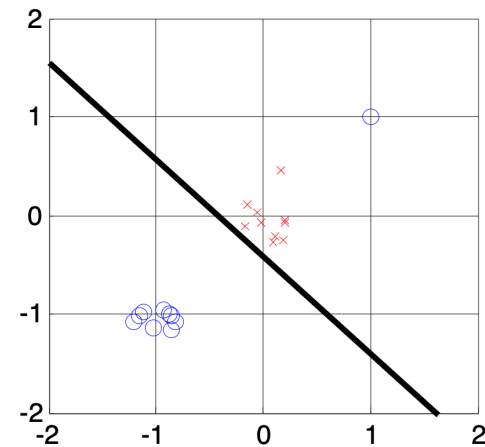


More examples

Unevenly distributed
training data



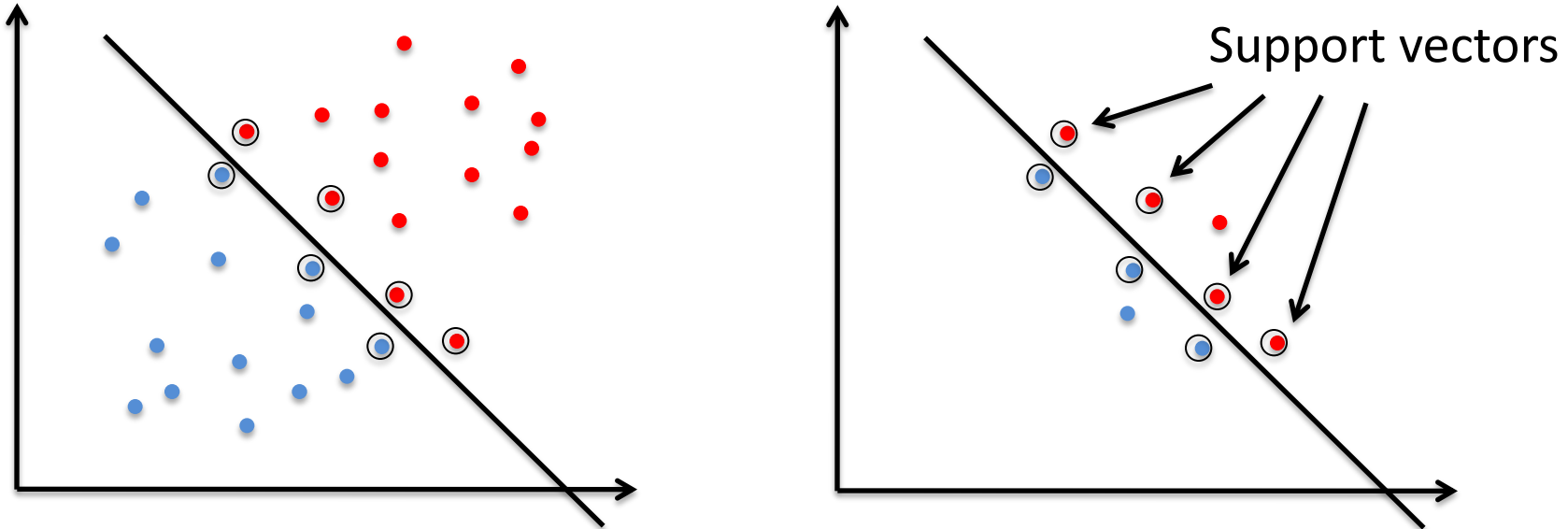
Outlier



$$\mathcal{E}(\mathbf{w}) = \sum_{i=1}^N (\mathbf{w}^T \mathbf{x}_i - y_i)^2$$

Support Vector Machines (SVM)

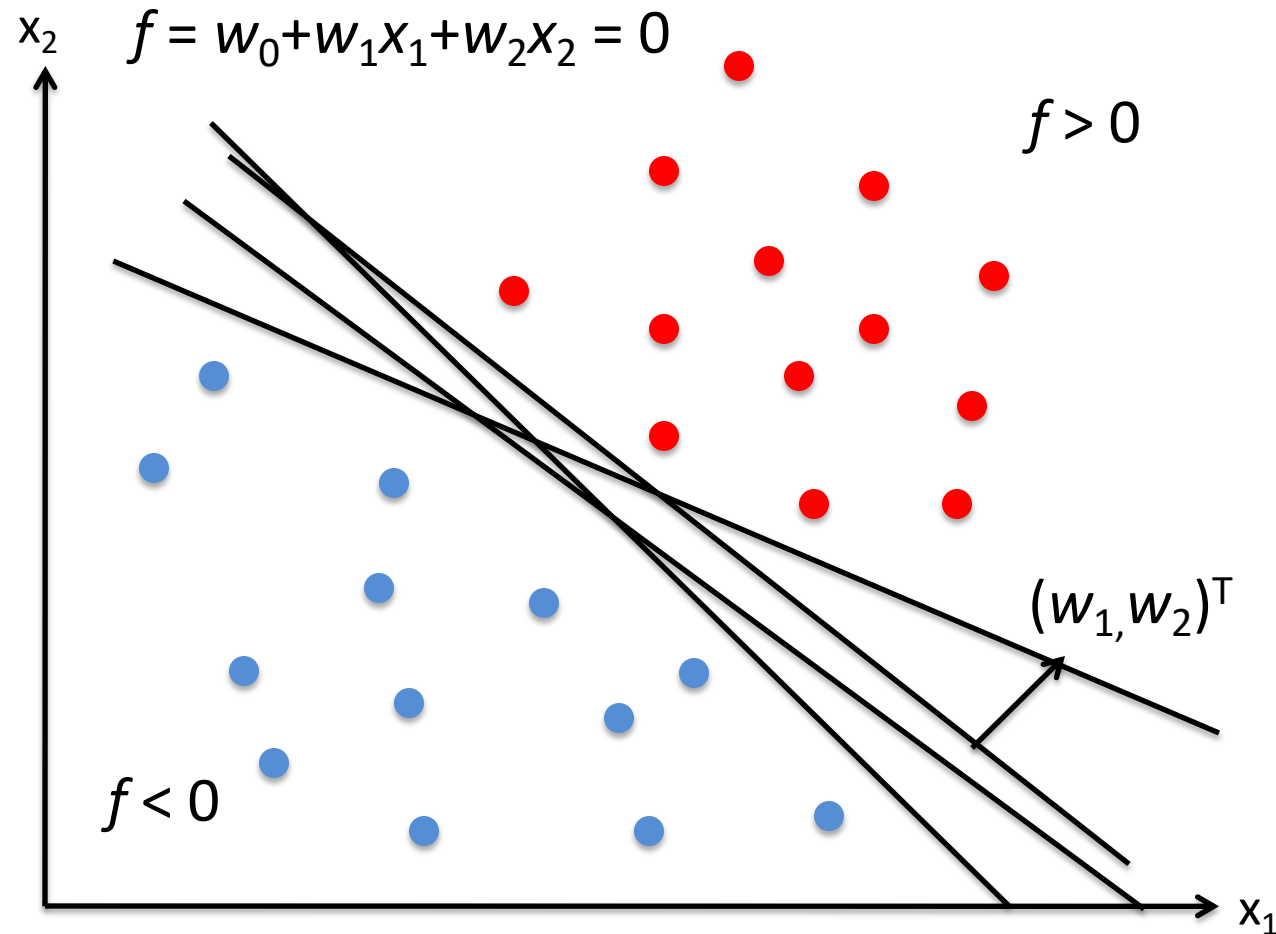
Idea!



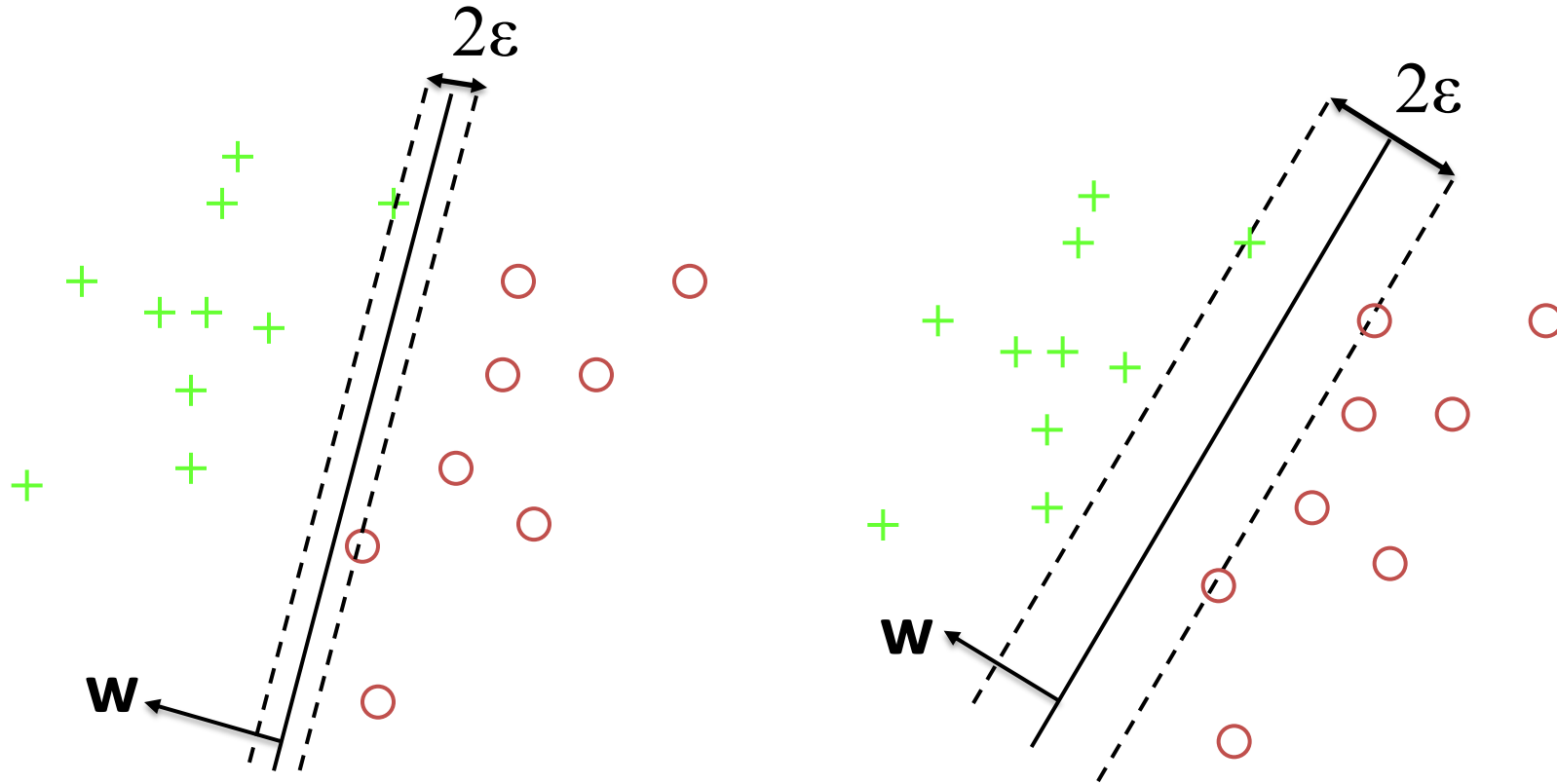
Optimal separation line remains the same, feature points close to the class limits are more important!

These are called *support vectors*!

Which linear classifier to choose?



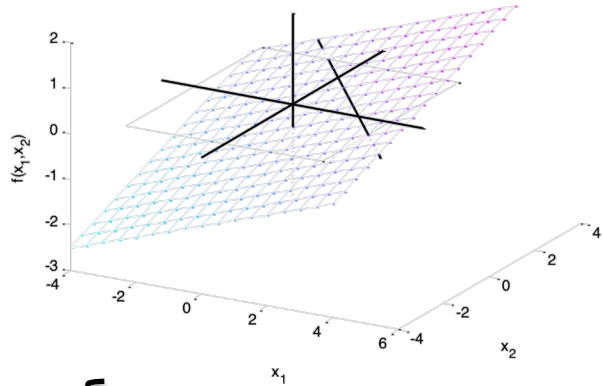
SVM – Maximum margin



Choose w that gives maximum margin ϵ !

SVM – Loss function

Scaling of \mathbf{w} is free – Pick arbitrary sample \mathbf{x}_s as support vector and choose scaling so that $\mathbf{w}^T \mathbf{x}_s + w_0 = 1$!

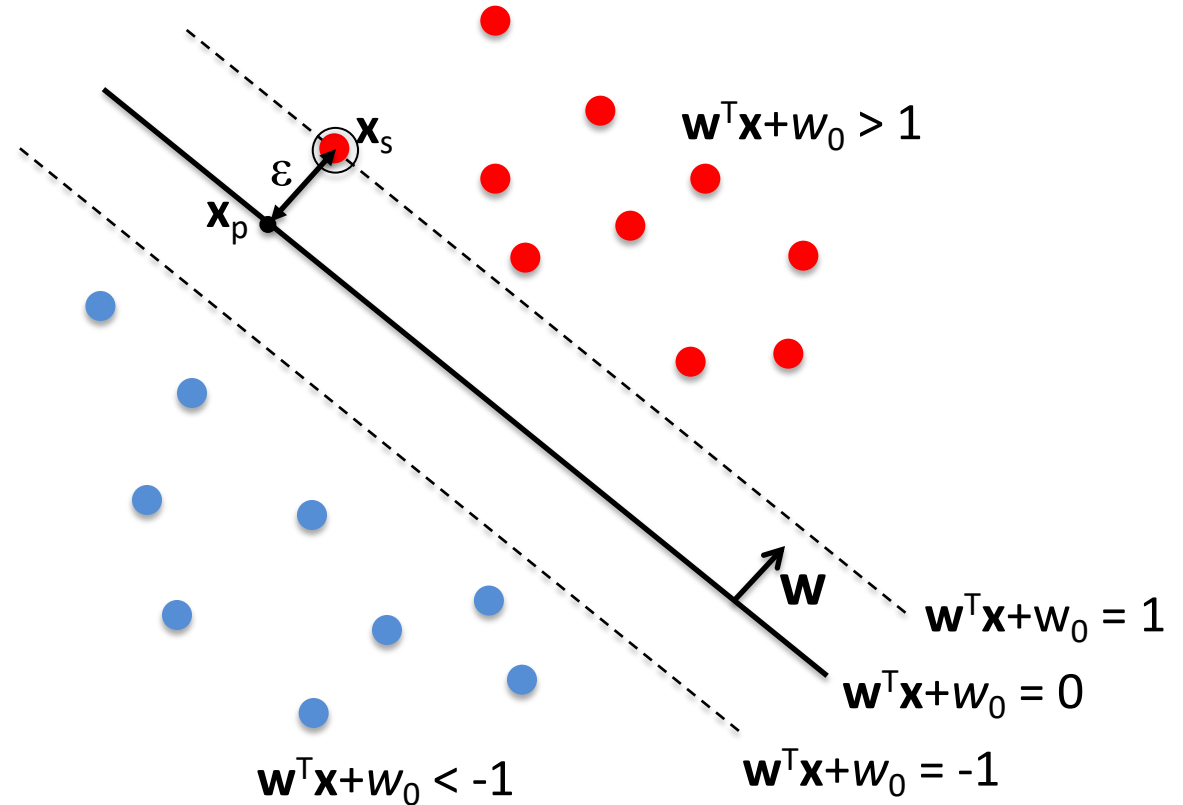


$$\begin{cases} \mathbf{w}^T \mathbf{x}_s + w_0 = 1 \\ \mathbf{x}_s = \mathbf{x}_p + \varepsilon \hat{\mathbf{w}} \end{cases}$$

$$\mathbf{w}^T (\mathbf{x}_p + \varepsilon \hat{\mathbf{w}}) + w_0 = 1$$

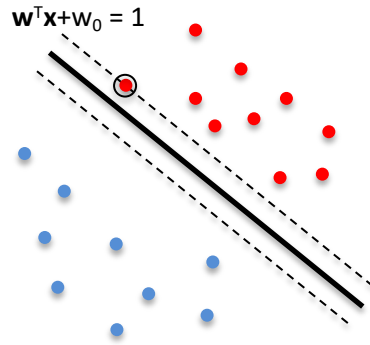
$$\varepsilon \underbrace{\mathbf{w}^T \hat{\mathbf{w}}}_{\|\mathbf{w}\|} + \underbrace{\mathbf{w}^T \mathbf{x}_p + w_0}_0 = 1$$

$$\|\mathbf{w}\| \hat{\mathbf{w}}^T \hat{\mathbf{w}} = \|\mathbf{w}\|$$



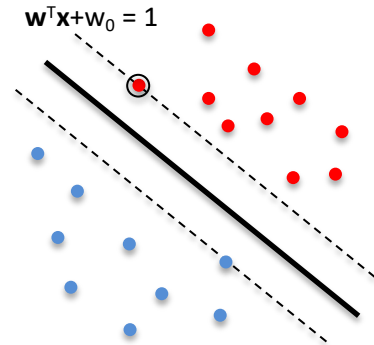
For the chosen support vector, $\varepsilon(\mathbf{w}) = 1 / \|\mathbf{w}\|$

SVM loss function examples



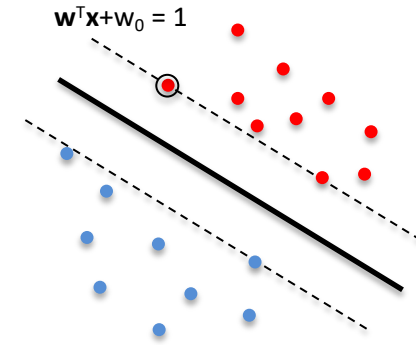
Example 1:

Boundary not in middle \rightarrow
Large $\|w\|$ (steep function) \rightarrow
Small margin $\epsilon(w) = 1 / \|w\|$



Example 2:

Boundary more in middle \rightarrow
Smaller $\|w\|$ (flatter function) \rightarrow
Larger margin $\epsilon(w) = 1 / \|w\|$



Example 3:

Tilt boundary somewhat \rightarrow
Smallest possible $\|w\|$ \rightarrow
Largest margin $\epsilon(w) = 1 / \|w\|$

Choosing another training sample as reference support vector can give an even larger margin!

SVM – Loss function, cont.

Maximizing $\varepsilon = 1 / \|\mathbf{w}\|$ is the same as minimizing $\|\mathbf{w}\|^2$!

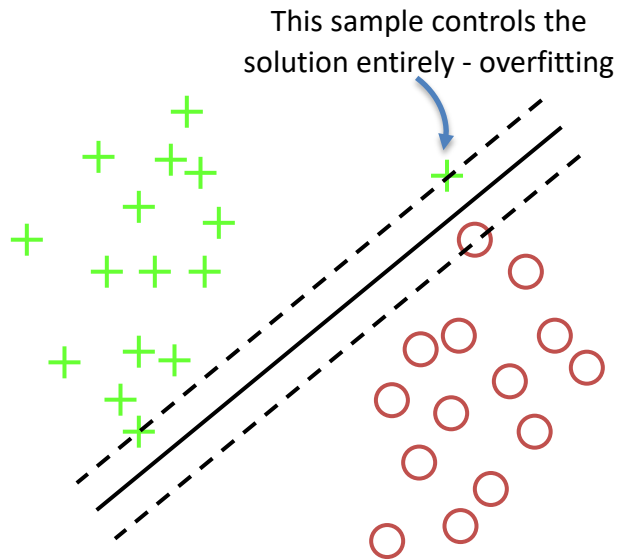
$$\min \|\mathbf{w}\|^2$$

$$\text{subject to } y_i (\mathbf{w}^T \mathbf{x}_i + w_0) \geq 1$$

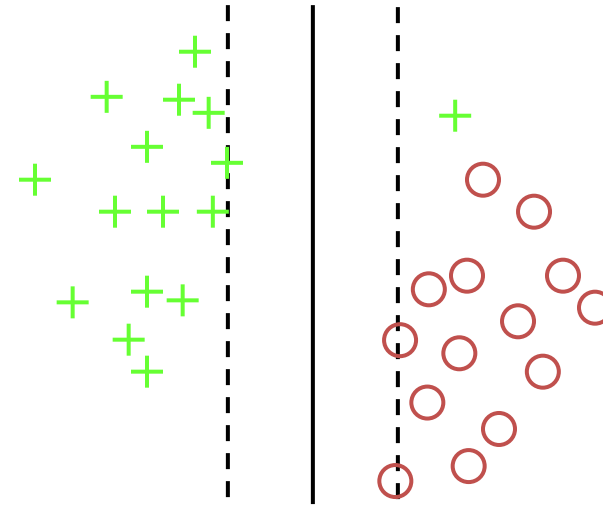
No training samples must reside in the margin region!

Optimization procedure outside the scope of this course...

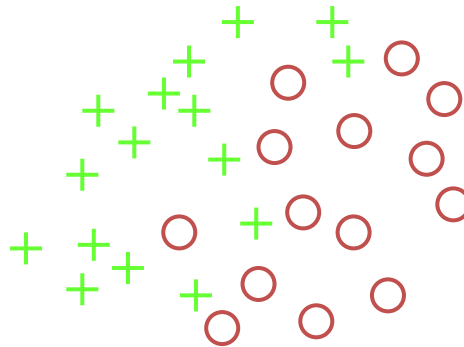
SVM – Soft margin



Would this be a better boundary?



What if the training data is not separable with a line?



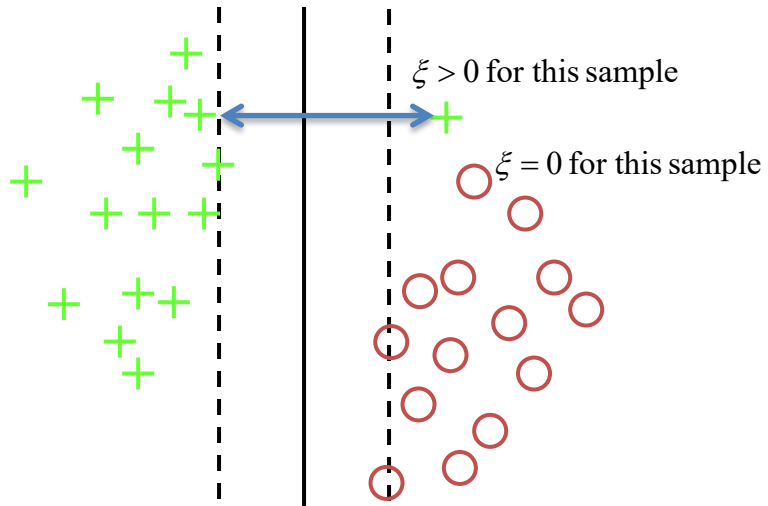
SVM – Soft margin, cont.

$$\min_{\mathbf{w}, w_0, \xi_i} \left\{ \|\mathbf{w}\|^2 + C \sum_{i=1}^K \xi_i \right\}$$

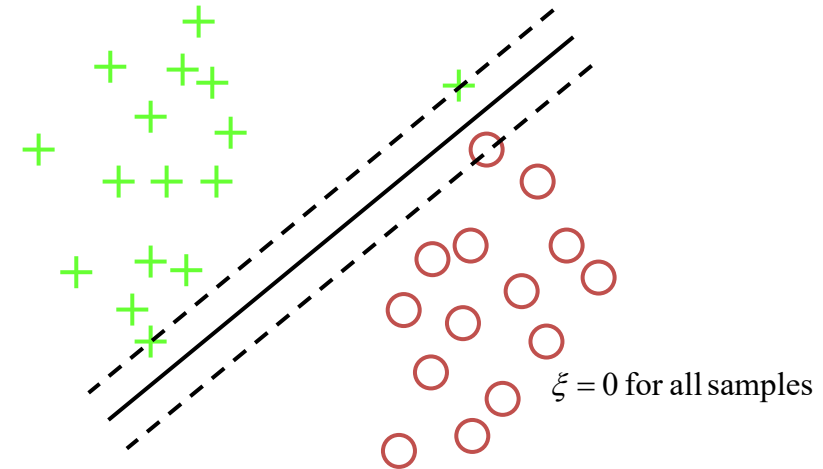
user-defined trade-off parameter

slack variable

$$\text{subject to } y_i(\mathbf{w}^T \mathbf{x}_i + w_0) \geq 1 - \xi_i$$



Solution for small C



Solution for large C

SVM – Choosing C

Solve the optimization problem with different C :s and choose the solution with highest accuracy according to cross-validation procedure.

$$C = 2^{-5}, 2^{-3}, \dots, 2^{15}$$

Training data	Training data	Validation data
Training data	Validation data	Training data
Validation data	Training data	Training data

Practical guide:

<http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf>

Summary – Linear classifiers

- **Different loss functions give different algorithms**
- **Square error loss**
 - Sensitive to outliers and training data distribution when applied as in this lecture.
 - Improvements possible (lecture 3).
 - Local minima.
- **Support Vector Machines (maximum margin loss)**
 - Non-linear extension possible (lecture 9).
 - No local minima.
- **Linear Discriminant Analysis (Lecture 8)**
 - Simple to implement, very useful as a first classifier to try