# Lecture 7

## Classification II
## KNN, SVM, Neural networks

**Golnaz Taheri, PhD**
Senior Lecturer, Stockholm University

# Outline

- Eager vs Lazy learning
- The nearest neighbor classifier
- The Perceptron
- Support Vector Machine (SVM)
- Logistic regression


Stockholms universitet

# Classification recap

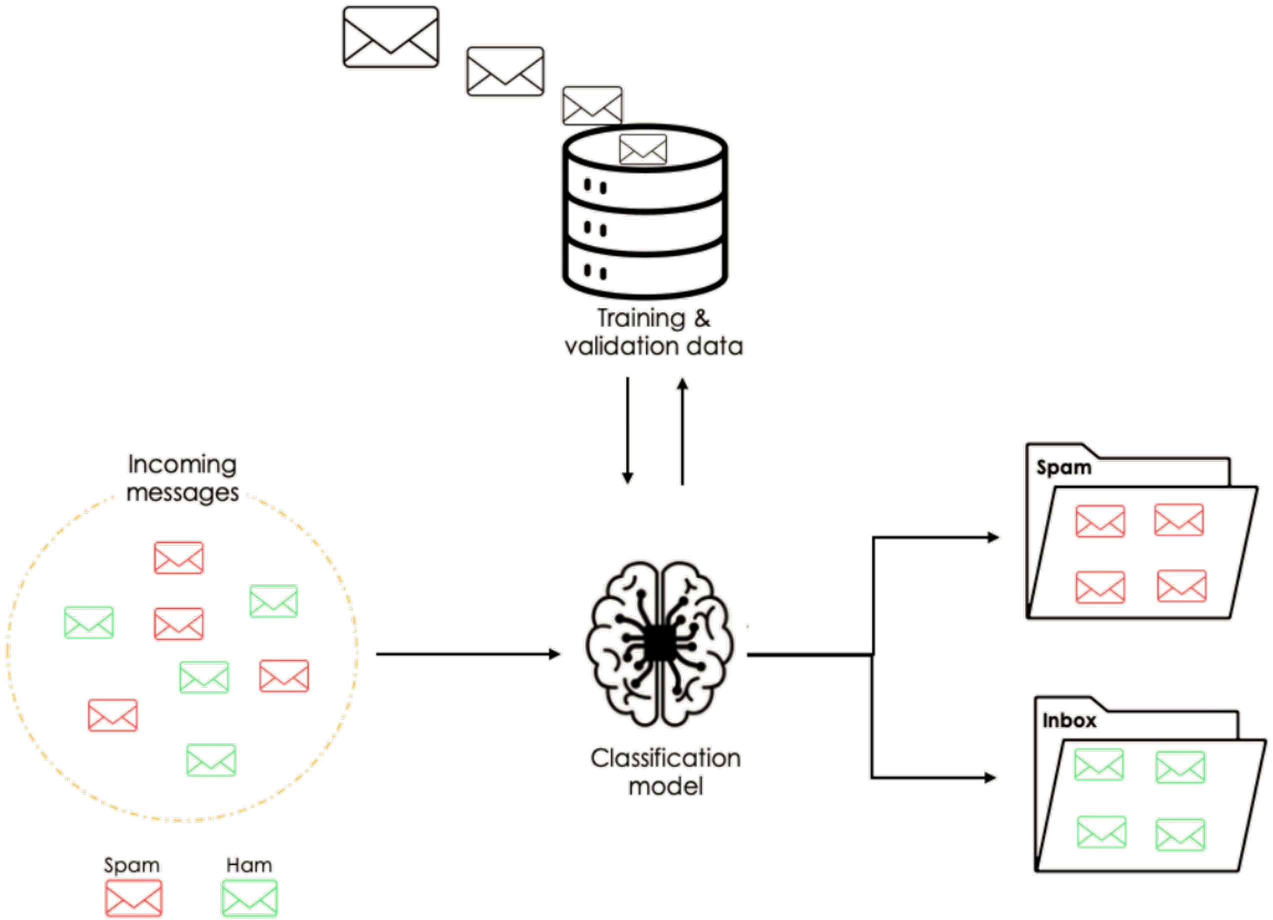
- Classification is a supervised machine learning method where the model tries to predict the correct label of a given input data.

- In classification, the model is fully trained using the training data, and then it is evaluated on test data before being used to perform prediction on new unseen data.


Stockholms universitet

# Classification recap

- An algorithm can learn to predict whether a given email is spam or no spam.






Stockholms universitet

# Eager Learning

- Eager learning methods construct general, explicit description of the target function based on the provided training examples.

≡ *one-fits-all* ≡ *input independent*

- Learn the model as soon as the training data becomes available

- More training time, less prediction time.
  - Support vector machines (SVM)
  - Decision tree
  - Artificial neural networks

# Lazy Learning

- Lazy learning methods simply store the data and generalizing beyond these data is postponed until an explicit request is made.
- Delay model-building until testing data needs to be classified
- Less training time, more prediction time.
- The model itself consists of the training data

# K-Nearest Neighbor Classification

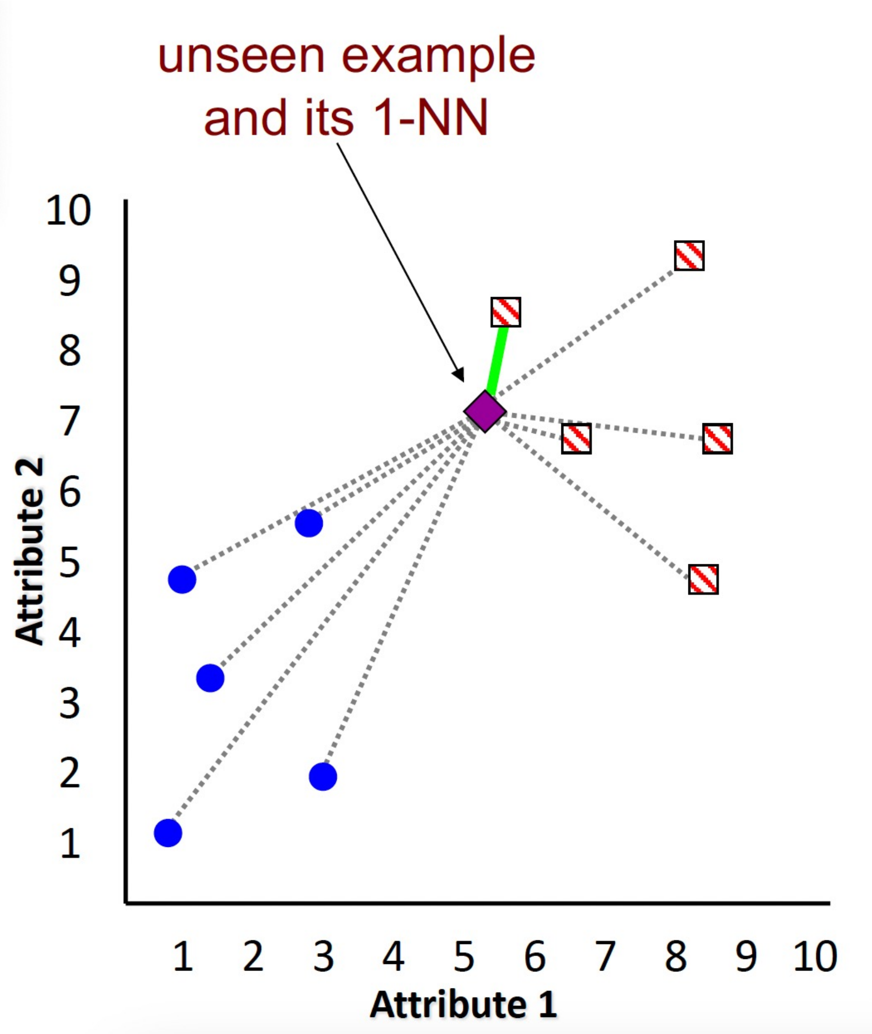
- KNN algorithm is one of the simplest classification algorithm

- Non-parametric
  - it does not make any assumptions on the underlying data distribution

- Lazy learning algorithm.
  - Means that the training phase is pretty fast .
  - Lack of generalization means that KNN keeps all the training data.

- Its purpose is to use a database in which the data points are separated into several classes to predict the classification of a new sample point.

# Nearest Neighbor Classifier




Goal: build no model, just find the most similar training example(s)

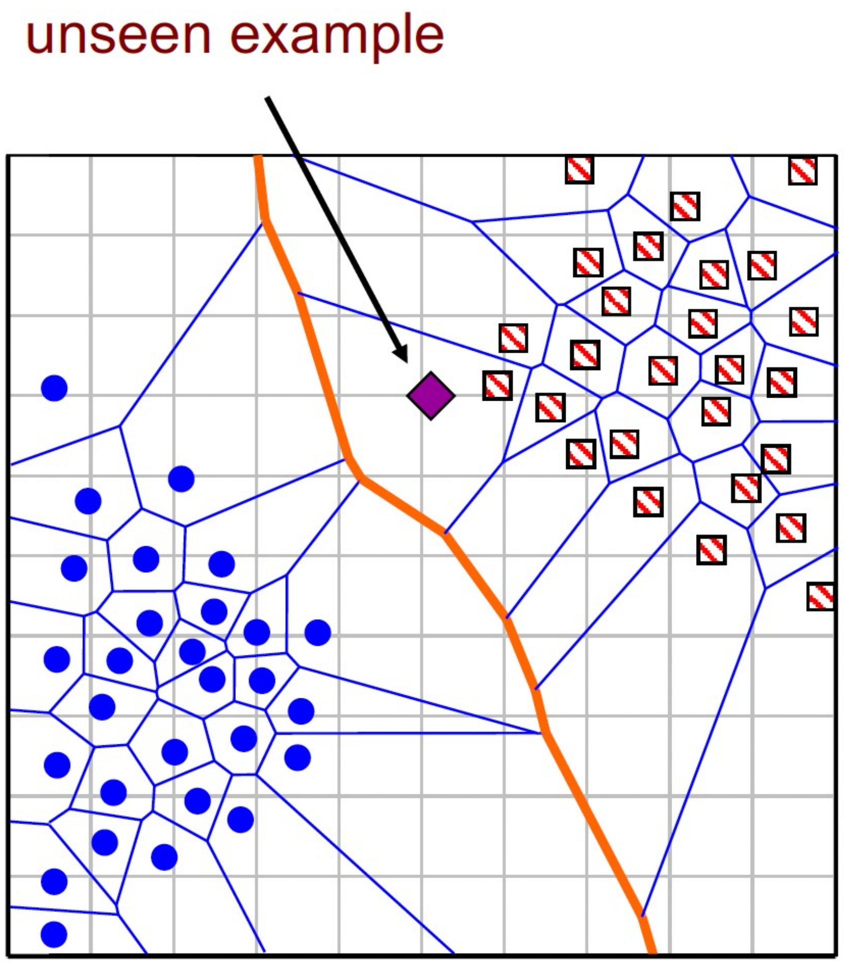


Stockholms universitet

# The decision boundary of 1-NN

- Decision boundaries: surfaces that simply divide the space into regions “belonging “ to each Instance

- For 1-NN it is the **Voronoi Diagram** of the training set

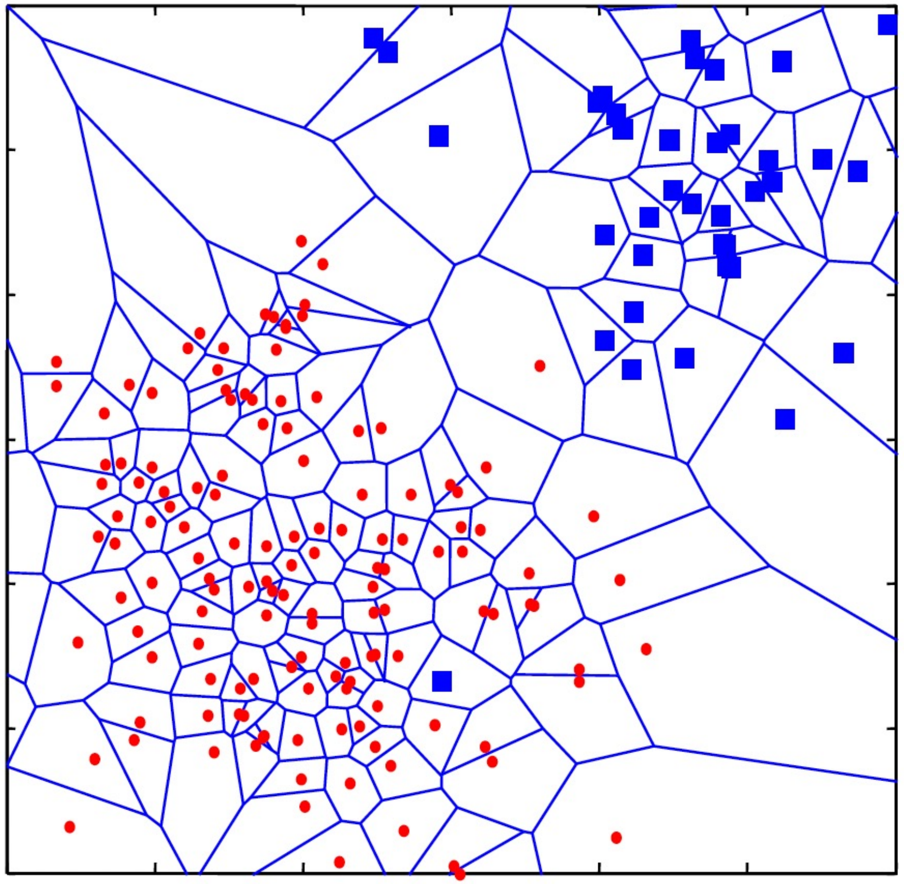


# 1-NN is sensitive to outliers

Solution: K-NN classifier:

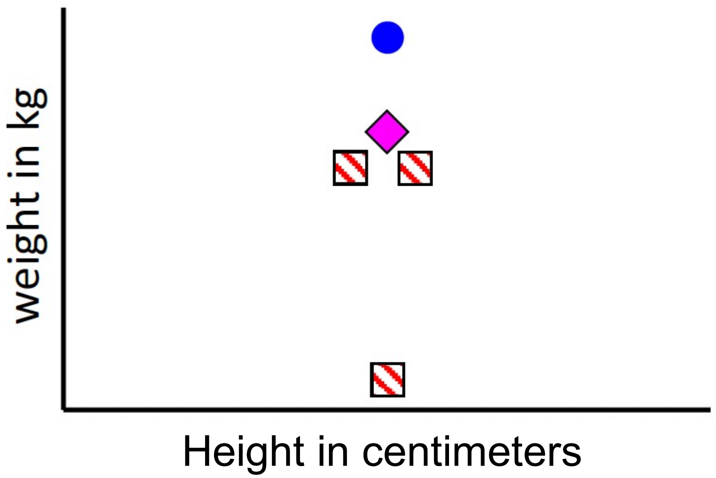
- We measure the distance to the nearest K instances, and let them vote

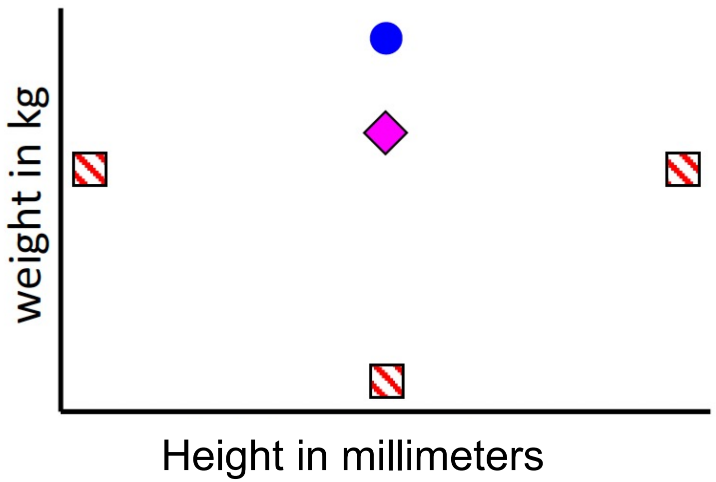
- K is typically chosen to be an odd number



Stockholms universitet

# 1-NN is sensitive to the units of measurement




X axis measured in **centimeters**
Y axis measure in kg
The nearest neighbor to the pink unknown instance is red

X axis measured in **millimeters**
Y axis measure in kg
The nearest neighbor to the pink unknown instance is blue

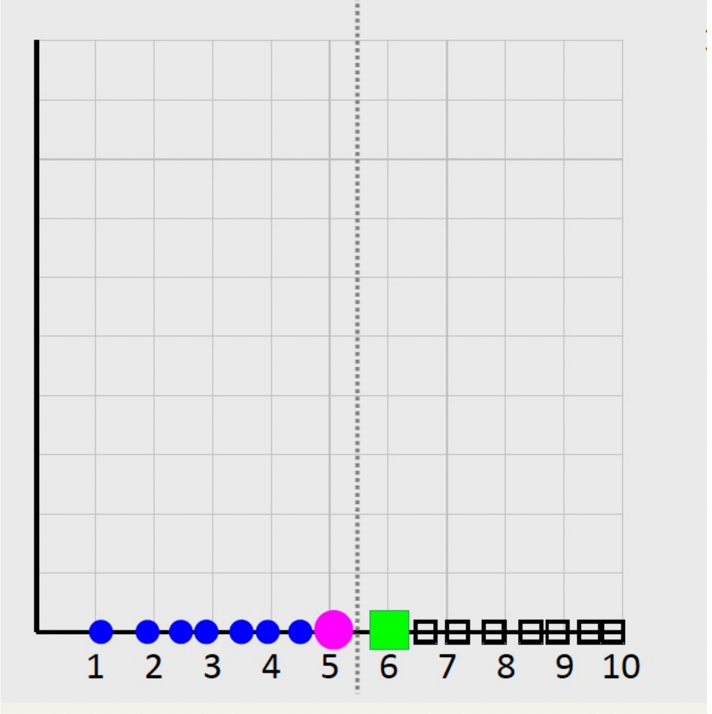
Solution: normalize the units per attribute

- min-max normalization
- standardization

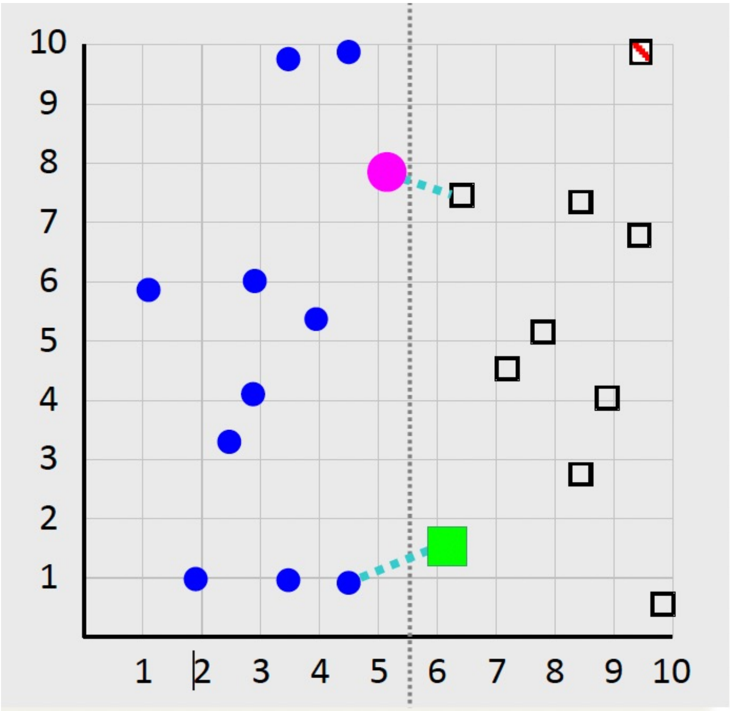


Stockholms universitet

# 1-NN is sensitive to irrelevant attributes




1 attribute
Can provide perfect classification



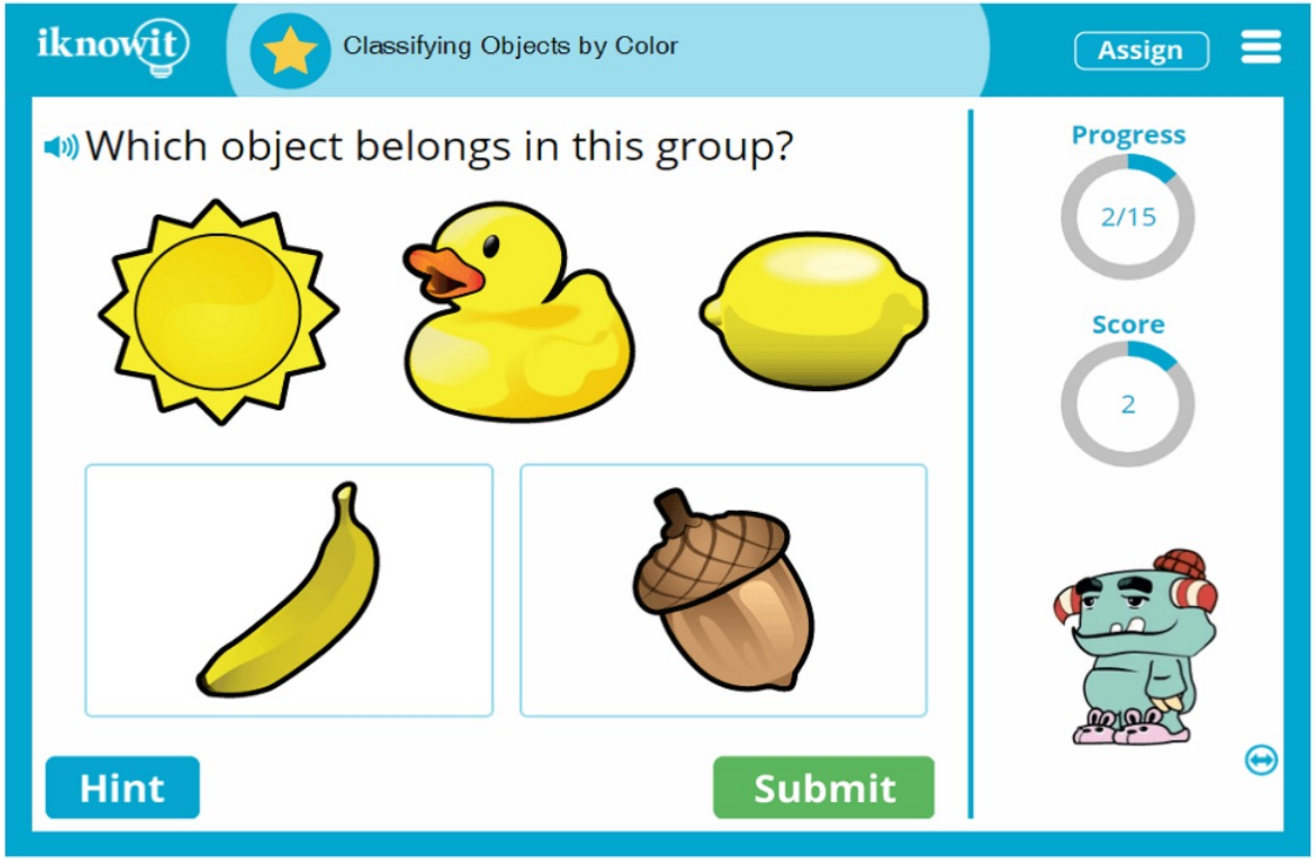

2 attributes
Classification is wrong!

Solution: add more training data or ask domain experts what features are important

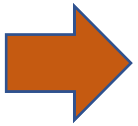
# Characteristics of K-NN classifiers

- Lazy learners: computational time in classification, no model building
- Eager learners: computational time in model building
- Decision trees try to find global models, K-NN classifiers take into account local information, i.e., example-based information, by looking at the neighborhood of the test example
- K-NN classifiers depend a lot on the choice of distance measure

# Distance measure importance




| Color |
|---|
| Red |
| Green |
| Blue |
| Yello |
| Purple |

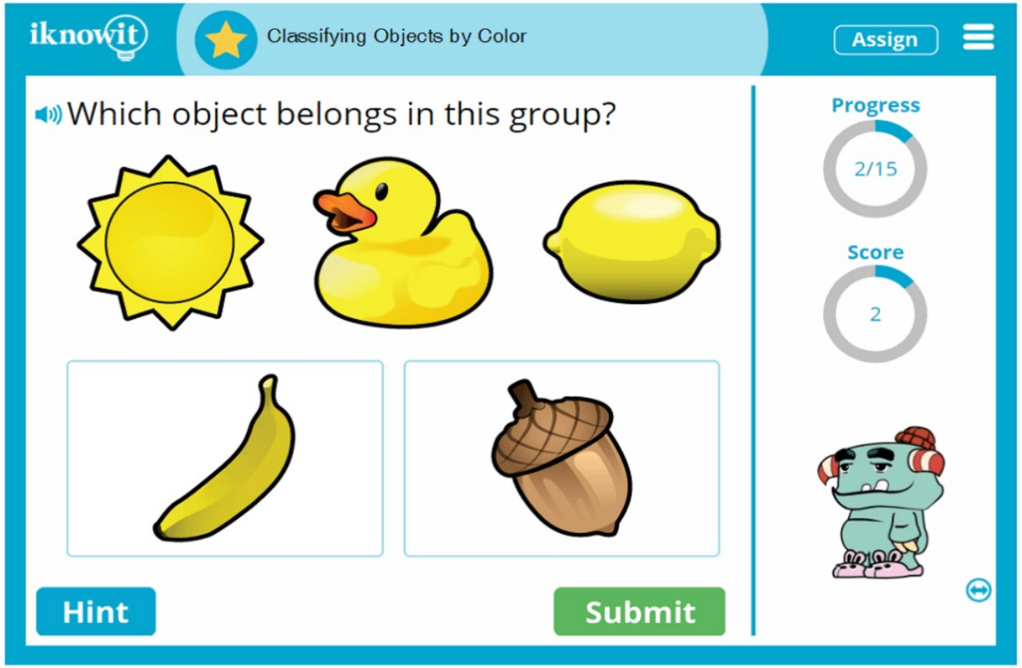


| Color |
|---|
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |



Stockholms universitet

# Distance measure importance




| Color |
|---|
| Red |
| Green |
| Blue |
| Yello |
| Purple |

| Red | Green | Blue | Yellow | Purple |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 |

# The Perceptron

- Input: each example $x$ has a set of attributes $x = \{a_1, a_2, \dots, a_m\}$ and is of class $y$

- Estimated classification output: $u$

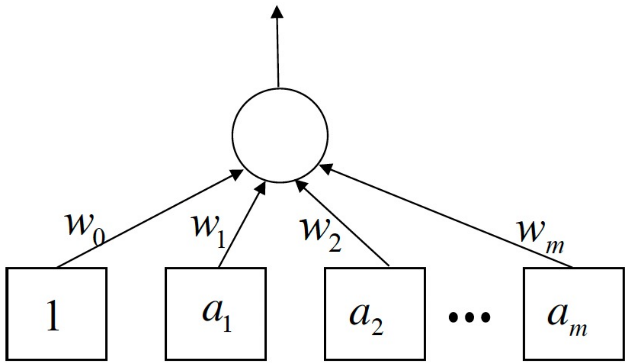
- Task: express each sample $x$ as a weighted combination (linear combination) of the attributes


Stockholms universitet

# The Perceptron

- Define a set of $m$ weights: $< w_1, w_2, \ldots, w_m >$
- Multiply each attribute with their weight and sum them up
- Use an additional weight $w_0$
  - intercept value, makes the model more general

$$f(x) = w_1 a_1 + w_2 a_2 + \ldots + w_m a_m + w_0 = \underbrace{< w, x >}_{\text{inner or dot product of } w \text{ and } x} + w_0$$




$$f(x) > 0 \rightarrow u_i = 1$$

$$f(x) \leq 0 \rightarrow u_i = -1$$

How do you find the weights ?

# Online Perceptron Algorithm

Let $\mathbf{w} \leftarrow (0,0,0,\ldots,0)$

Repeat

Accept training example $i : (\mathbf{x}_i \ , \ y_i)$

$u_i \leftarrow \mathbf{w} \cdot \mathbf{x}_i$

if $y_i \cdot u_i \ \leq \ 0$

$\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$



Stockholms universitet

# The Perceptron: example

4 examples with 4 attributes

- Suppose our training set contains 4 examples with 3 attributes:

| α1 | α2 | α3 | Class Y |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | -1 |
| 0 | 1 | 1 | -1 |

- We want to learn a straight line (hyperplane in this case) that separates them:

$$f(x) = \langle w, x \rangle + w_0 = w_1\alpha_1 + w_2\alpha_2 + w_3\alpha_3 + w_0$$


Stockholms universitet

# The Perceptron: example

Let $\mathbf{w} \leftarrow (0,0,0,\ldots,0)$

Repeat

Accept training example $i : (\mathbf{x}_i \, , \, y_i)$

$u_i \leftarrow \mathbf{w} \cdot \mathbf{x}_i$

if $y_i \cdot u_i \leq 0$

$\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$

| W0 | W1 | W2 | W3 | | α0 | α1 | α2 | α3 | y | f(x) | class |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | $x_1$ | 1 | 0 | 0 | 0 | 1 | 0 | → -1 |
| | | | | | 1 | 0 | 1 | 0 | 1 | | |
| | | | | | 1 | 1 | 0 | 1 | -1 | | |
| | | | | | 1 | 0 | 1 | 1 | -1 | | |

| W0 | W1 | W2 | W3 |
|---|---|---|---|
| 1 | 0 | 0 | 0 |

$$f(x) = \langle w, x \rangle + w_0 = w_1\alpha_1 + w_2\alpha_2 + w_3\alpha_3 + w_0$$

$$f(x) \leq 0 \rightarrow u_i = -1 \qquad f(x) > 0 \rightarrow u_i = 1$$



Stockholms universitet

# The Perceptron: example

Let $\mathbf{w} \leftarrow (0,0,0,\ldots,0)$

Repeat

Accept training example $i$ : $(\mathbf{x}_i\ ,\ y_i)$

$u_i \leftarrow \mathbf{w} \cdot \mathbf{x}_i$

if $y_i \cdot u_i \ \leq\ 0$

$\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$

| | W0 | W1 | W2 | W3 | | α0 | α1 | α2 | α3 | y | f(x) | class |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 0 | 0 | 0 | | 1 | 0 | 0 | 0 | 1 | 1 | → 1 |
| | | | | | | 1 | 0 | 1 | 0 | 1 | 1 | → 1 |
| | | | | | $x_3$ | 1 | 1 | 0 | 1 | -1 | 1 | → 1 |
| | | | | | | 1 | 0 | 1 | 1 | -1 | | |

| W0 | W1 | W2 | W3 |
|---|---|---|---|
| 0 | -1 | 0 | -1 |

$$f(x) = \langle w, x \rangle + w_0 = w_1\alpha_1 + w_2\alpha_2 + w_3\alpha_3 + w_0 = 0$$

$$f(x) \leq 0 \rightarrow u_i = -1 \qquad f(x) > 0 \rightarrow u_i = 1$$



Stockholms universitet

# The Perceptron: example

Let $\mathbf{w} \leftarrow (0,0,0,\ldots,0)$

Repeat

  Accept training example $i : (\mathbf{x}_i\ ,\ y_i)$

  $u_i \leftarrow \mathbf{w} \cdot \mathbf{x}_i$

  if $y_i \cdot u_i\ \leq\ 0$

    $\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$

| W0 | W1 | W2 | W3 | | α0 | α1 | α2 | α3 | y | f(x) | class |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | | 1 | 0 | 0 | 0 | 1 | 1 | → 1 |
| | | | | | 1 | 0 | 1 | 0 | 1 | 1 | → 1 |
| | | | | $x_3$ | 1 | 1 | 0 | 1 | -1 | 1 | → 1 |
| | | | | | 1 | 0 | 1 | 1 | -1 | | |

| W0 | W1 | W2 | W3 |
|---|---|---|---|
| 0 | -1 | 0 | -1 |

$$f(x) = \langle w, x \rangle + w_0 = w_1\alpha_1 + w_2\alpha_2 + w_3\alpha_3 + w_0 = 0$$

$$f(x) \leq 0 \rightarrow u_i = -1 \qquad f(x) > 0 \rightarrow u_i = 1$$



Stockholms universitet

# The Perceptron: example

Let $\mathbf{w} \leftarrow (0,0,0,\ldots,0)$

Repeat

Accept training example $i : (\mathbf{x}_i \,,\, y_i)$

$u_i \leftarrow \mathbf{w} \cdot \mathbf{x}_i$

if $y_i \cdot u_i \;\le\; 0$

$\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$

| W0 | W1 | W2 | W3 | | α0 | α1 | α2 | α3 | y | f(x) | class |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | -1 | 0 | -1 | $x_1$ | 1 | 0 | 0 | 0 | 1 | **0** | **→ -1** |
| | | | | | 1 | 0 | 1 | 0 | 1 | | |
| | | | | | 1 | 1 | 0 | 1 | -1 | | |
| | | | | | 1 | 0 | 1 | 1 | -1 | -1 | → -1 |

| W0 | W1 | W2 | W3 |
|---|---|---|---|
| 1 | -1 | 0 | -1 |

$$f(x) = \langle w, x \rangle + w_0 = w_1\alpha_1 + w_2\alpha_2 + w_3\alpha_3 + w_0 = 0$$

$$f(x) \le 0 \rightarrow u_i = -1 \qquad f(x) > 0 \rightarrow u_i = 1$$

# The Perceptron: example

$f(x)$ can also be written as a linear combination of all or some of the ***training examples***

Let $\mathbf{w} \leftarrow (0,0,0,\ldots,0)$

Repeat

Accept training example $i : (\mathbf{x}_i \ , y_i)$

$u_i \leftarrow \mathbf{w} \cdot \mathbf{x}_i$

if $y_i \cdot u_i \ \leq \ 0$

$\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$

| W0 | W1 | W2 | W3 | α0 | α1 | α2 | α3 | y | f(x) | class |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | -1 | 0 | -1 | 1 | 0 | 0 | 0 | 1 | 1 | → 1 |
| | | | | 1 | 0 | 1 | 0 | 1 | 1 | → 1 |
| | | | | 1 | 1 | 0 | 1 | -1 | -1 | → -1 |
| | | | | 1 | 0 | 1 | 1 | -1 | -0 | → -1 |

$$f(x) = w_1\alpha_1 + w_2\alpha_2 + w_3\alpha_3 + w_0 = -\alpha_1 - \alpha_3 + 1$$

$$f(x) = \sum_i k_i y_i \langle x_i, x \rangle + b$$

$$k_1 = 2 \quad k_2 = 0 \quad k_3 = 1 \quad k_4 = 0$$

$$f(x) = \langle w, x \rangle + w_0 = w_1\alpha_1 + w_2\alpha_2 + w_3\alpha_3 + w_0 = 0$$

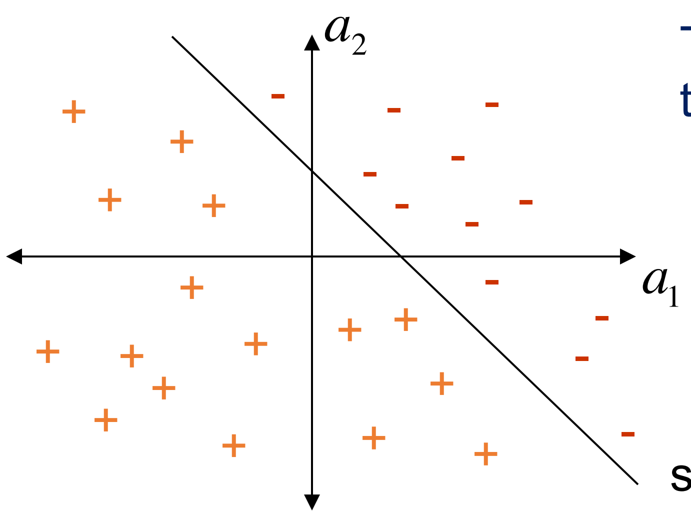
$$f(x) \leq 0 \rightarrow u_i = -1 \qquad f(x) > 0 \rightarrow u_i = 1$$



Stockholms universitet

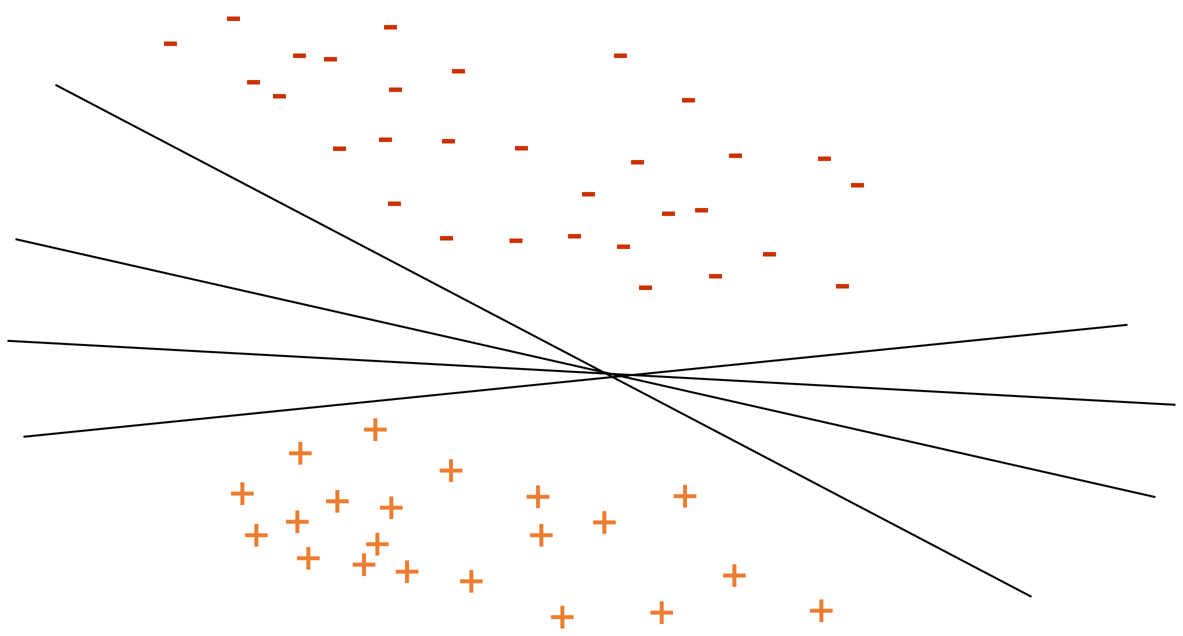
# The Perceptron

$$w_0 + w_1 a_1 + w_2 a_2 + \ldots + w_m a_m = 0$$



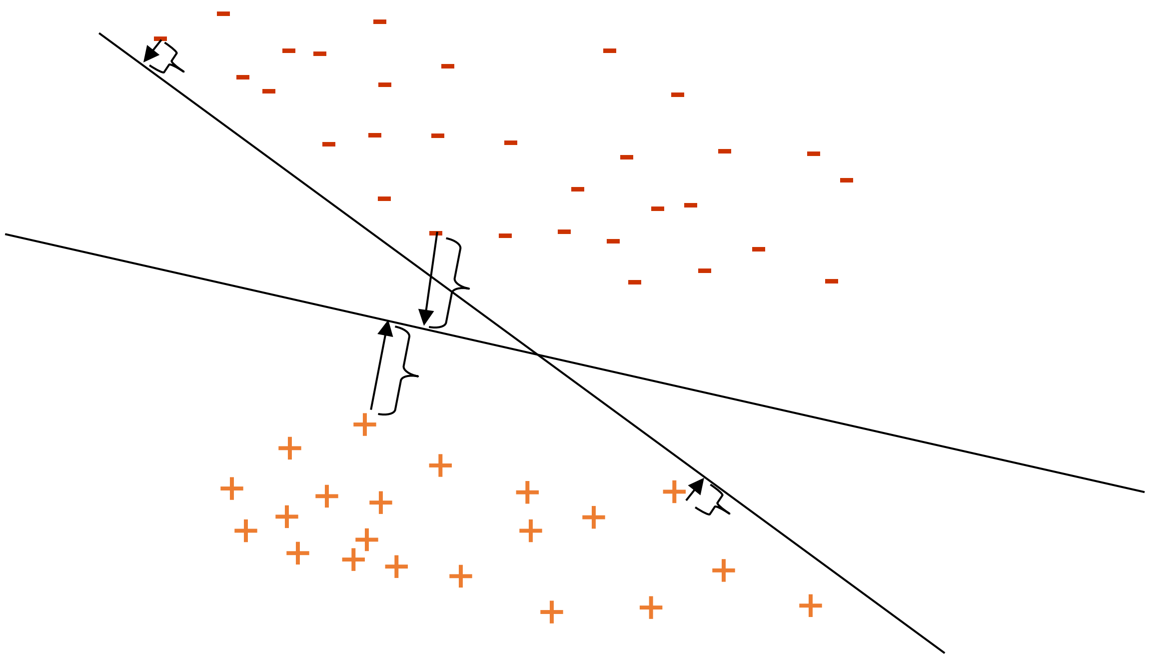

The perceptron learning algorithm is guaranteed to find a separating hyperplane – if there is one.

# Many possible separating hyperplanes



Which hyperplane to choose?

# Choosing hyperplanes

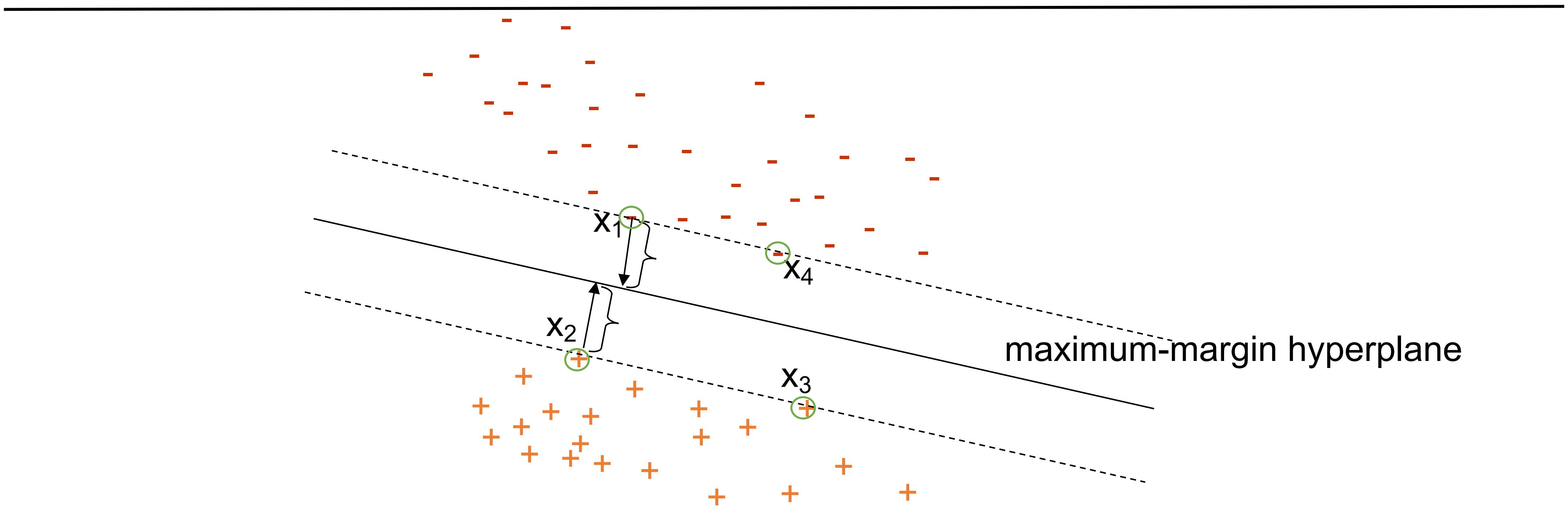


Margin: defined by the two points (one from the '+' set and the other from the '–' set) with the minimum distance to the separating hyperplane



Stockholms universitet

# Linear SVM




$x_1$, $x_2$, $x_3$, and $x_4$ are the **support vectors**, i.e., the closest points to the margins from both sides



Stockholms universitet

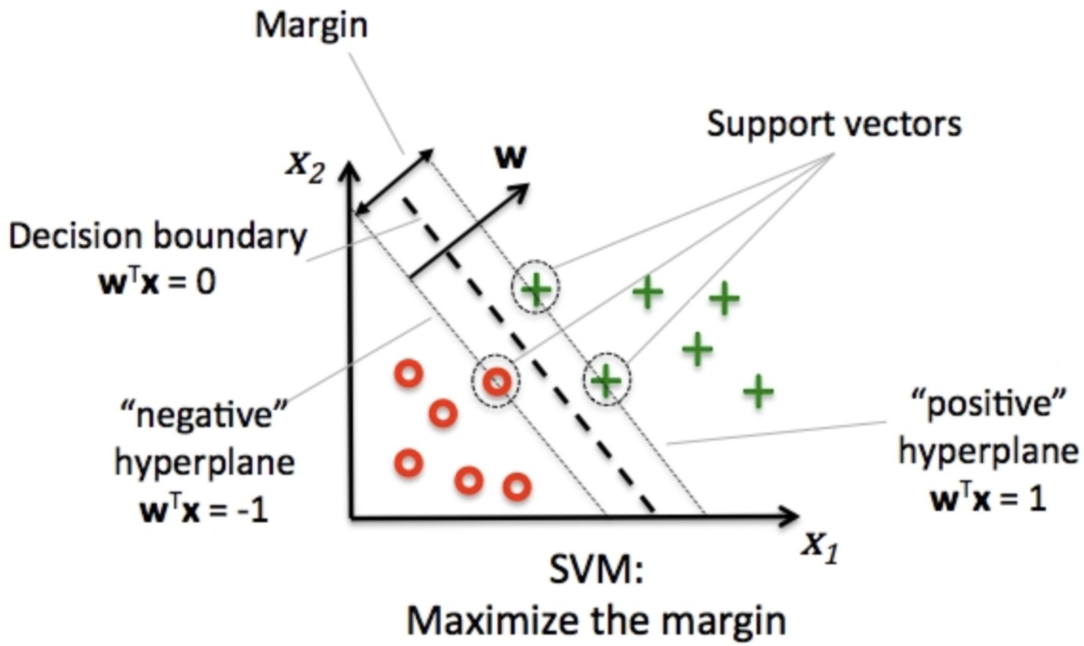
# The maximum margin hyperplane

- One reasonable choice for the separating hyperplane:
    - The hyperplane that represents the largest separation, or ***margin***, between the two classes
    - We choose the hyperplane so that the distance from it to the nearest data point(s) on each side is maximized
    - If such a hyperplane exists, it is known as the **maximum-margin hyperplane**
    - This is a **quadratic programming problem** that can be easily solved analytically by standard methods (optimization using Lagrange multipliers)

# The maximum margin hyperplane

- It is desirable to design linear classifiers that maximize the margins of their decision boundaries

- This ensures that their worst-case generalization errors are minimized




**Risk of overfitting is reduced by finding the maximum margin hyperplane!**



Stockholms universitet

# The maximum margin hyperplane

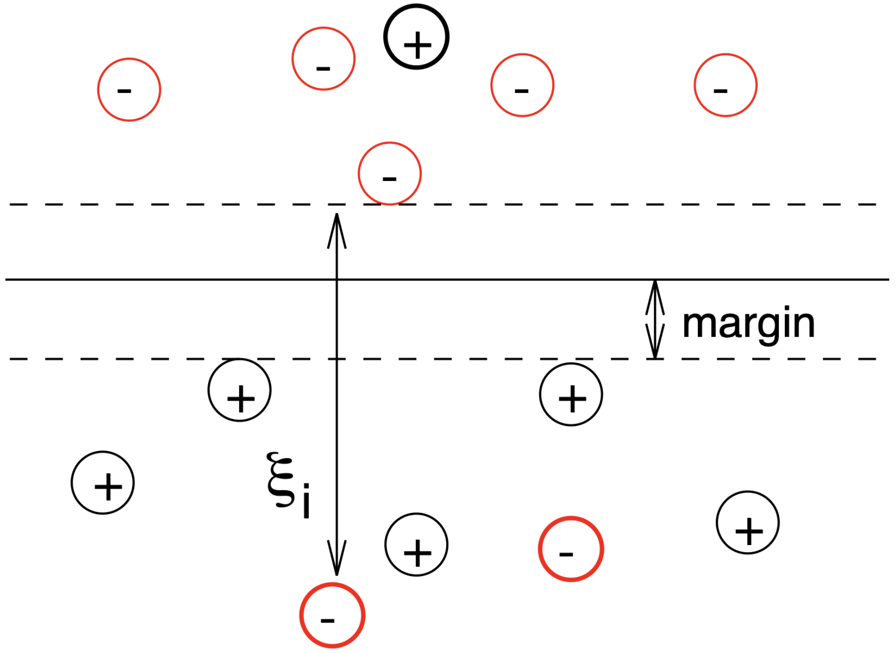
- **Linear support vector machines:**
  - Find the maximum-margin hyperplane
  - Express this hyperplane as a linear combination of the data points $(x_i, y_i)$:

$$f(x) = \sum_i k_i y_i \langle x_i, x \rangle + b$$

  - The points (from each side) with the smallest distance from the hyperplane are called ***support vectors***
  - **Formally:** support vectors are the elements of the training set that would change the position of the dividing hyperplane if removed
  - We want to express $f(x)$ as a linear combination of the support vectors $\boldsymbol{x_i}$, by identifying them and learning their corresponding weights $\boldsymbol{k_i}$

# Overfitting and slack variables

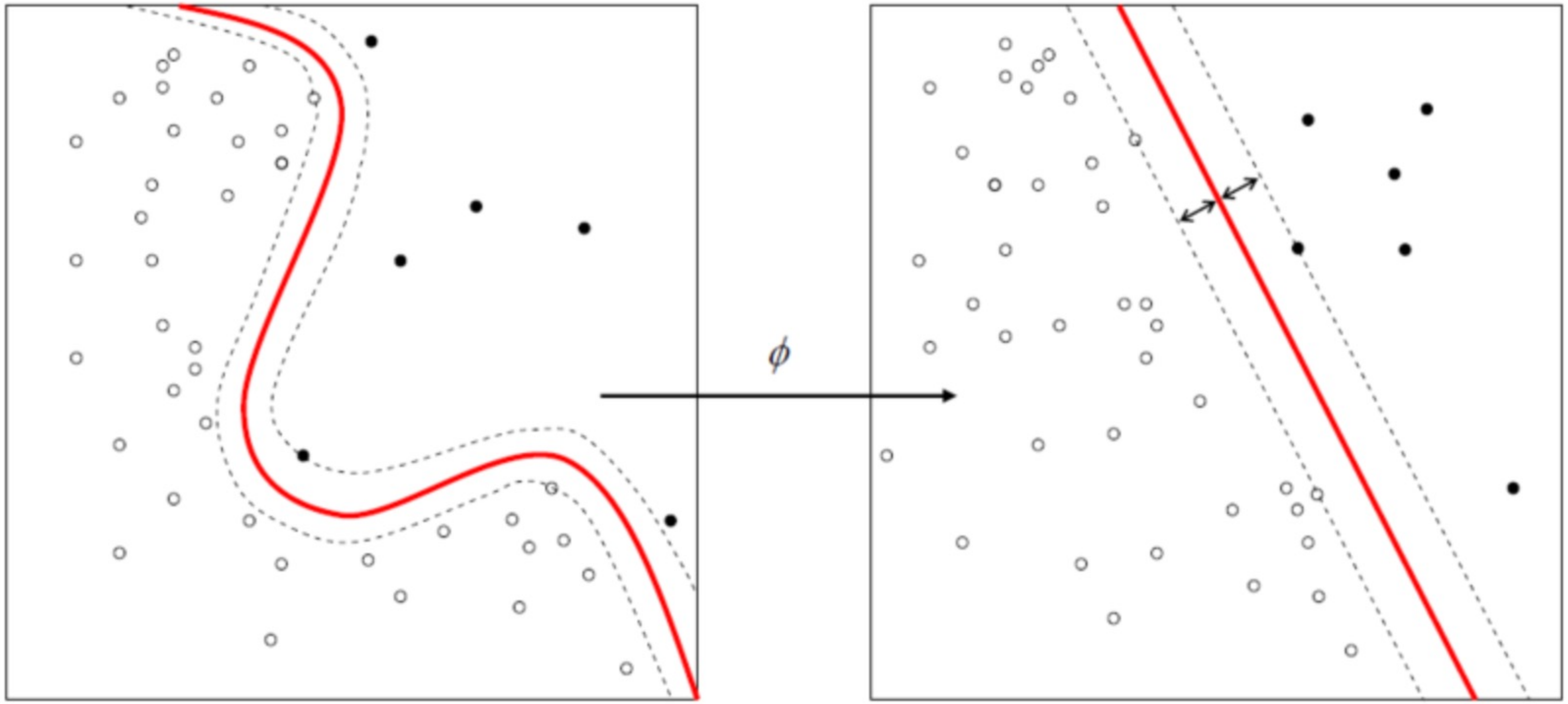
- **Curved decision boundary:** it is not always very likely to find a line dividing the training examples
- Data points may be noise or outliers
- **Alternative:** introduce a smooth boundary that ignores a few data points rather than defining a very curved boundary
- This is handled by introducing **slack variables $\xi_i$** that measure the distance of a point to its marginal hyperplane




**Prevention of overfitting!**

Stockholms universitet

# What if the classes are not linearly separable?




Define a *mapping* $\varphi(x)$ that maps each attribute of point $x$ to a new space where points are *linearly separable*

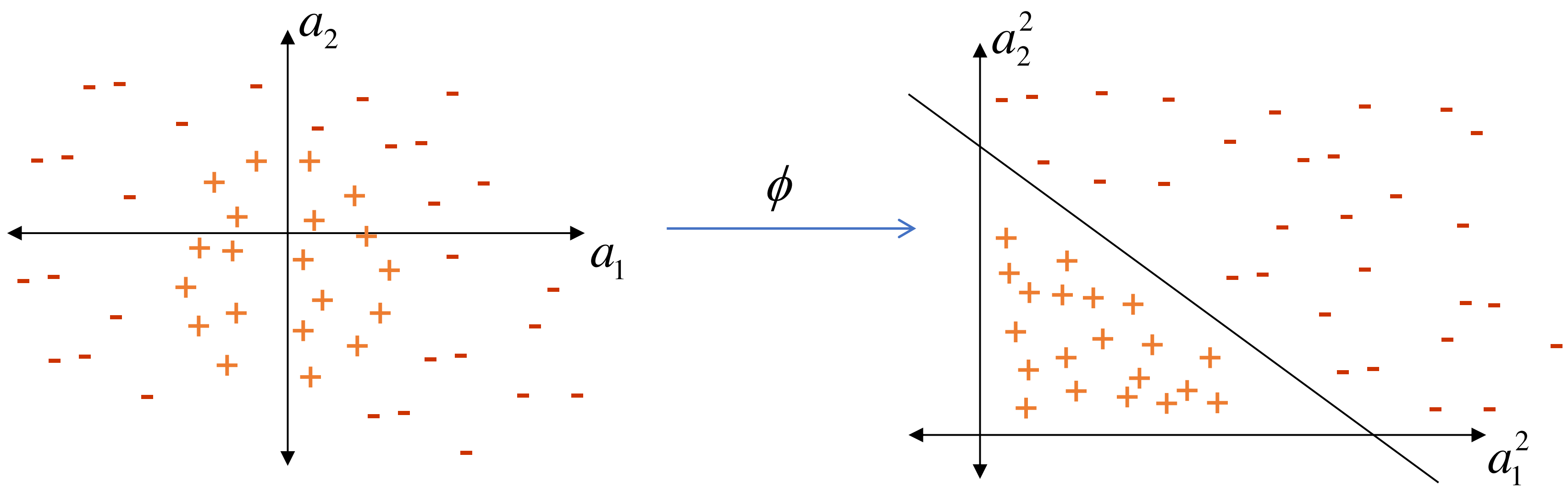


Stockholms universitet

# What if the classes are not linearly separable?

- Original attributes are mapped to a new space to obtain linear separability

$$f_{orig}(x) = \langle w, x \rangle + w_0 = \sum_i k_i y_i \langle x_i, x \rangle + b \qquad f(x) = \sum_i k_i y_i \langle \varphi(x_i), \varphi(x) \rangle + b$$




- **Another alternative: Artificial Neural Networks – deep learning**

# Kernels

- Assume two points in the original space:

  $x_1 = (\alpha_1, \alpha_2)$ and $x_2 = (b_1, b_2)$

**kernel function:** $K(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$

A function that returns the value of the dot product between the images $\varphi$ of two data points

**Complex mapping:**

- $\varphi(x)$ can be quite complex, for example: $\varphi(x) = (a_1^2, a_2^2, \sqrt{2}a_1a_2)$

- Hence:

$$\varphi(x_1) = \varphi((a_1, a_2)) = (a_1^2, a_2^2, \sqrt{2}a_1a_2) \qquad \varphi(x_2) = \varphi((b_1, b_2)) = (b_1^2, b_2^2, \sqrt{2}b_1b_2)$$

$$\langle \varphi(x_1), \varphi(x_2) \rangle = \left\langle (a_1^2, a_2^2, \sqrt{2}a_1a_2), (b_1^2, b_2^2, \sqrt{2}b_1b_2) \right\rangle \longleftarrow \text{COMPLEX}$$

$$= (a_1^2b_1^2 + a_2^2b_2^2 + 2a_1b_1a_2b_2)$$

$$= (a_1b_1 + a_2b_2)^2$$

$$= \langle (a_1, a_2), (b_1, b_2) \rangle^2 = \langle x_1, x_2 \rangle^2 \longleftarrow \text{SIMPLE}$$



Stockholms universitet

# Kernels

**Observation:**

- $\varphi(x)$ can be quite complex, for example see below

**kernel function:**

$$K(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$$

**Kernel trick:**

- Do not need to actually compute the vectors $\varphi$ in the mapped space
- Just need to identify a simple form of the dot product of the mapped values $\varphi(x_1)$ and $\varphi(x_2)$

$$\varphi(x_1) = \varphi((a_1, a_2)) = (a_1^2, a_2^2, \sqrt{2}a_1 a_2) \qquad \varphi(x_2) = \varphi((b_1, b_2)) = (b_1^2, b_2^2, \sqrt{2}b_1 b_2)$$

$$\boxed{\langle \varphi(x_1), \varphi(x_2) \rangle = \left\langle (a_1^2, a_2^2, \sqrt{2}a_1 a_2), (b_1^2, b_2^2, \sqrt{2}b_1 b_2) \right\rangle} \longleftarrow \text{COMPLEX}$$

$$= (a_1^2 b_1^2 + a_2^2 b_2^2 + 2a_1 b_1 a_2 b_2)$$

$$= (a_1 b_1 + a_2 b_2)^2$$

$$\boxed{= \langle (a_1, a_2), (b_1, b_2) \rangle^2 = \langle x_1, x_2 \rangle^2} \longleftarrow \text{SIMPLE}$$

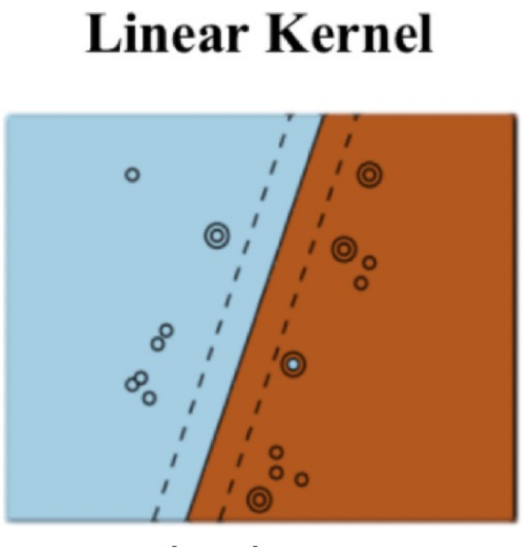


Stockholms universitet

# Beyond the Hyperplane, Choose your kernel

- **Linear (dot) kernel**
  - This is linear classifier, use it as a test of non-linearity
  - Or as a reference for the classification improvement with non-linear kernels

$$K(x_1,x_2)=\langle x_1,x_2\rangle^1$$



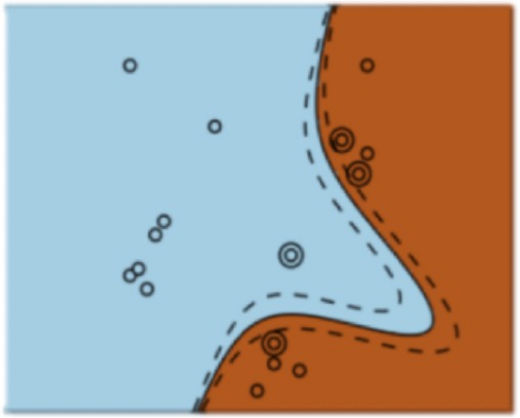

Standard SVM with Hyperplane Boundary

# Beyond the Hyperplane, Choose your kernel

- **Polynomial**
  - Simple, efficient for non-linear relationships
  - Identifies/exploits polynomial relationships between the variables
  - *d* – degree, high *d* leads to overfitting

$$K(x_1, x_2) = \langle x_1, x_2 \rangle^d$$

**Polynomial Kernel**



More flexible, but more potential for overfitting

# Kernels

- **Several Types of Kernels:**

$$K(x_1, x_2) = \langle x_1, x_2 \rangle^d$$

polynomial kernels

$$K(x_1, x_2) = \left(\langle x_1, x_2 \rangle + 1\right)^d$$

$$K(x_1, x_2) = e^{-\|x_1 - x_2\|^2 / 2\sigma^2}$$

Gaussian radial basis function

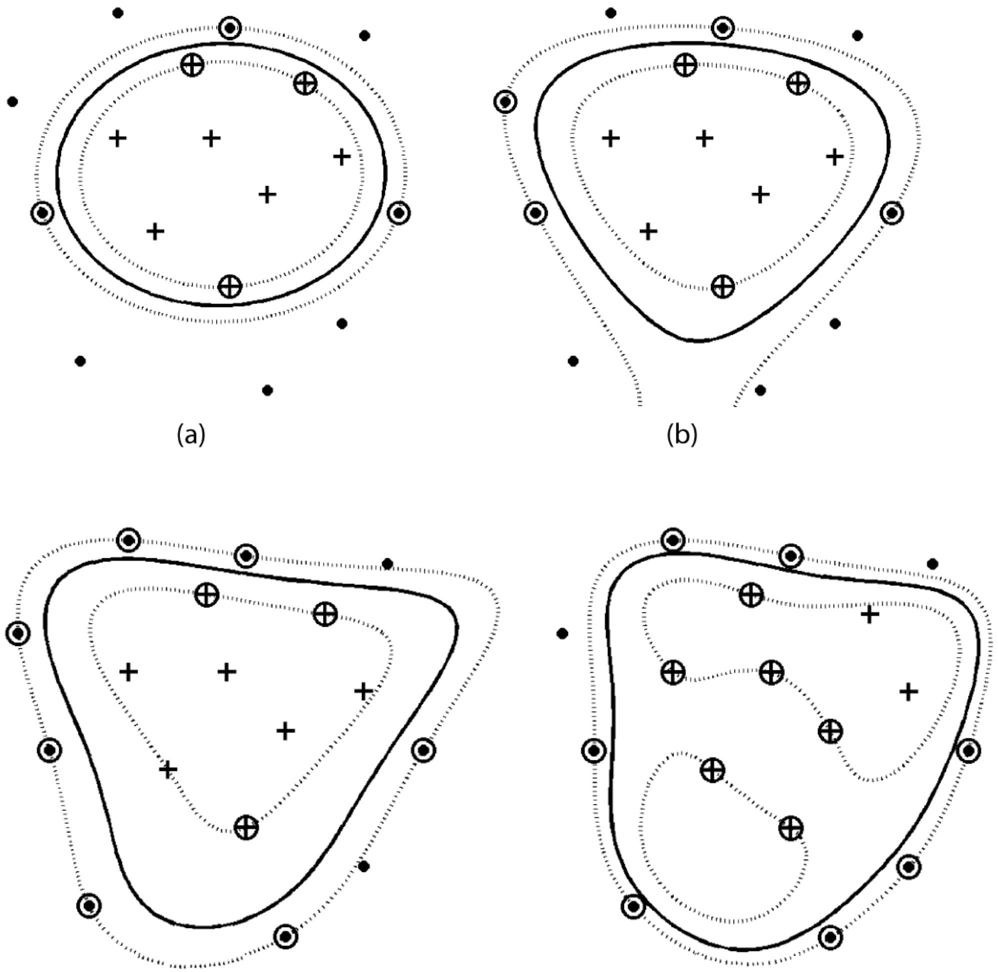
$$K(x_1, x_2) = \tanh\left(\kappa \langle x_1, x_2 \rangle - \delta\right)$$

two-layer sigmoidal neural network

# Polynomial kernel: overfitting as d increases!

$d = 2$ $d = 5$

$d = 3$ $d = 10$




Stockholms universitet

# Which kernel?

- So which kernel and which parameters should I use?
- The answer is data-dependent
- Several kernels should be tried
- Try the linear kernel first
- Check if classification can be improved with polynomial kernels
- Then try other nonlinear kernels (tradeoff between quality of the kernel and the number of dimensions)
- Select kernel + learn parameters

# Logistic regression

- The logistic regression algorithm finds the best **logistic function** that can describe the relationship between two variables:
  - dependent variable $\boldsymbol{y}$ (class variable)
  - independent variable(s) $\boldsymbol{X}$ (data variables)

- **Classic logistic regression:**
  - $\boldsymbol{y}$ is **binary**: i.e., it has two possible outcomes, e.g., win/loss, health/unhealthy
  - since $\boldsymbol{y}$ is binary, we often label classes as either 1 or 0



Stockholms universitet

# Computing the odds

- As new examples appear, we use the input variables and the logistic relationship to predict the **probability** $\boldsymbol{p}$ of a new example to belong to class $y = 1$:

$$P(y = 1 \mid X) = p$$

- Since $\boldsymbol{p}$ ranges between 0 and 1, we can convert this to a classification problem by using a *cutoff threshold*
- The higher the value of $\boldsymbol{p}$, the more likely the new example belongs to class $y = 1$, instead of $y = 0$
- For example, if we set the cutoff to $0.5$, this would mean that an example will be classified as $y = 1$, when $p > 0.5$, otherwise it will be classified as $y = 0$



Stockholms universitet

# Odds ratio

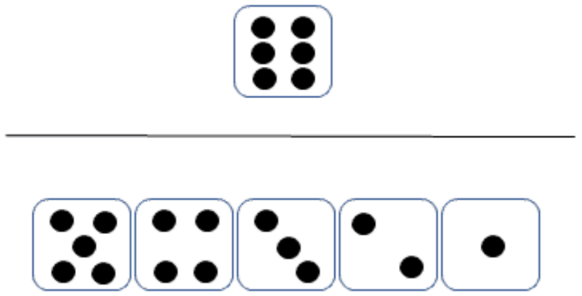
- Odds refer to the **ratio** of the probability of an event to happen divided by the probability of not happening
- It is a metric representing the likelihood of an event to occur

$$\text{odds} = \frac{\text{probability of something happening}}{\text{probability of something not happening}}$$

- Hence, the odds of the observation belonging to class $y = 1$ is $\boldsymbol{p/(1-p)}$:
  - When the odds are less than 1, they are against the example belonging to $y = 1$
  - When the odds are greater than 1, they are for the example belonging to $y = 1$

# Odds ratio

- For example, consider a fair six-sided dice:
  - The probability of a six coming up after a single roll is 1/6, and 5/6 of not happening
  - The odds in favor of winning are (1/6) / (5/6) = 1/5 or 1:5
  - The odds of losing are (5/6) / (1/6) = 5:1
  - The odds are clearly against winning



# The logit function

- The **log odds function** or **logit function** is $\log(p/(1-p))$, i.e., it corresponds to the logarithm of odds (the natural logarithm is most often used)

- **Ranges:**
  - $\boldsymbol{p}$ ranges from 0 to 1
  - $\boldsymbol{p}/(1-\boldsymbol{p})$ ranges from 0 to *infinity*
  - $\log(\boldsymbol{p}/(1-\boldsymbol{p}))$ ranges from *–infinity to infinity*



Stockholms universitet

# Logistic regression

- Assume a set of $\boldsymbol{m}$ variables $\{\alpha_1, \alpha_2, \dots, \alpha_m\}$ and a set of $\boldsymbol{m}$ weights $\{w_1, w_2, \dots, w_m\}$
- Similar to the ***Perceptron*** formulation, we want to express the logit function as a linear combination of these variables:

$$\boldsymbol{logit(p) = \log\left(\frac{p}{1-p}\right) = w_0 + w_1x_1 + w_1x_1 + \dots + w_mx_m}$$

- The **logistic function** is the inverse of the logit function above, i.e., we solve for $p$:

$$\boldsymbol{p = \frac{1}{1 + e^{-(w_0 + w_1a_1 + w_2a_2 + \dots + w_ma_m)}}}$$



Stockholms universitet

# The sigmoid function

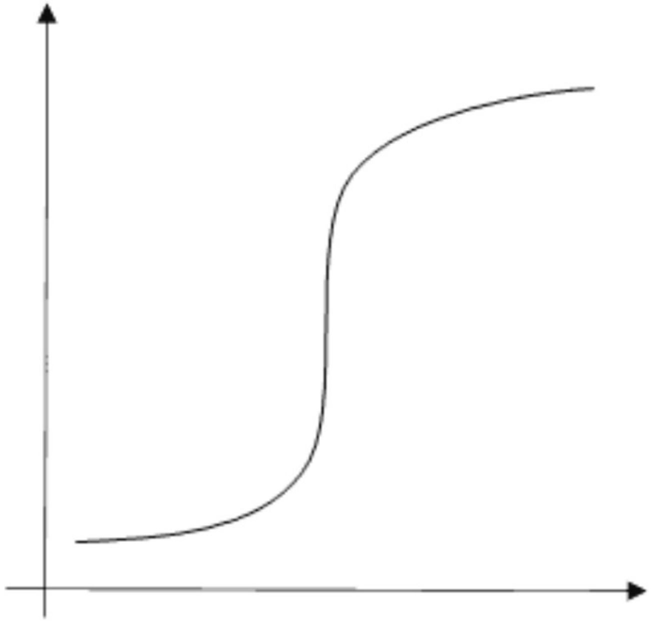
- The logistic function obtained above:

$$p = \frac{1}{1 + e^{-(w_0 + w_1 a_1 + w_2 a_2 + \ldots + w_m a_m)}}$$

is a type of a sigmoid function:

$$sigm(h) = \frac{1}{1 + e^{-h}}$$

$$h = w_0 + w_1 a_1 + w_2 a_2 + \ \ldots + w_m a_m$$



- The function ranges between 0 and 1

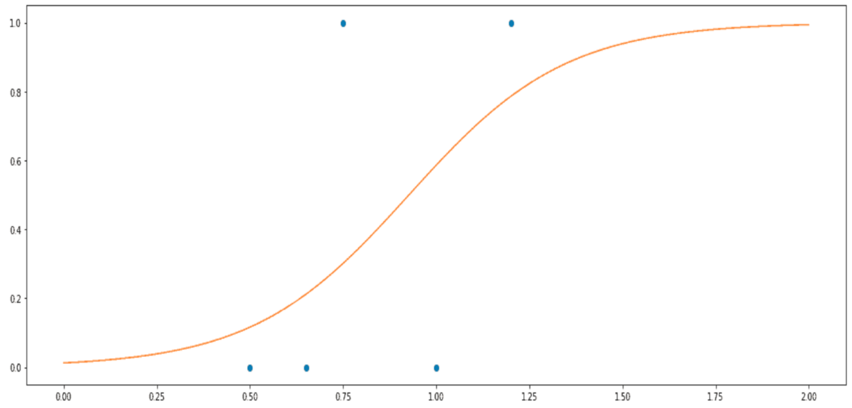
# Solving the logistic equation

- The goal is to find the values for the weights $\{w_0, w_1, w_2, \ldots, w_m\}$
- **Solution:** Maximum Likelihood Estimation (MLE)
- Use a likelihood function that measures how well a set of parameters fit a sample of data
- The parameter values that maximize the likelihood function are the maximum likelihood estimates
- Thus, the goal is to make inferences about the population that is most likely to have generated the training dataset
- **Assumption:** the data variables (attributes) are mutually independent

# Interpretation of logistic regression

- Suppose we have one variable and five examples

| Input x1 | Binary Output y |
|---|---|
| 0.5 | 0 |
| 1.0 | 0 |
| 0.65 | 0 |
| 0.75 | 1 |
| 1.2 | 1 |




- After solving the logistic equation, we get

$$\log(odds) = -4.411 + \mathbf{4.759}x_1$$

- This means that a one-unit increase of $x_1$, the log odds is expected to increase by **4.759**



Stockholms universitet

# Interpretation of logistic regression

- Suppose we have one variable and five examples

| Input x1 | Binary Output y |
|---|---|
| 0.5 | 0 |
| 1.0 | 0 |
| 0.65 | 0 |
| 0.75 | 1 |
| 1.2 | 1 |

Given an input example with $x_1 = 0.9$, we get

$$p = \frac{1}{1 + e^{-(-4.411+4.758*0.9)}} = \mathbf{46.8\%}$$

If our threshold is 0.5, then the predicted class is $\boldsymbol{y = 0}$

- After solving the logistic equation, we get

$$\log(odds) = -4.411 + \mathbf{4.759}x_1$$

- This means that a one-unit increase of $x_1$, the log odds is expected to increase by **4.759**



Stockholms universitet

# Types of logistic regression

- **Binary Logistic Regression:** the target variable has two possible categories; e.g., yes or no, spam or no spam, pass or fail
- **Multinomial Logistic Regression:** the target variable has three or more categories which are not in any particular order; e.g., categories of fruit: apple, mango, orange, and banana
- **Ordinal Logistic Regression:** the target variable has three or more ordinal categories; e.g., poor, average, good, very good, and excellent performance

# Assumptions of logistic regression

- The target variable should be binary, multinomial, or ordinal in nature
- The observations should be independent of each other; they should not come from repeated measurements
- There should be little or no multicollinearity among the independent variables; this means that the independent variables should not be too highly correlated with each other
- Linearity of the independent variables and log odds is assumed
- The success of Logistic Regression depends on the sample sizes; it requires a large sample size to achieve high predictive performance

# TODOs

**Reading:**

Main course book chapters:

8.2.3, 8.6,

11.1-11.3,

14.1-14.5

**Lab 3**

Sep 25

**Quiz 3**

# Coming up next

**Thursday**

Lab 2 – Clustering using Python

Lecture 8 – Classification III

**Friday**



Stockholms universitet

# Thanks!

golnaz.taheri@dsv.su.se