Machine Learning L+Pr

Béla J. Szekeres, PhD
Lecture 5

ELTE Faculty of Informatics, Szombathely, Hungary

Topics of this day

- Boosting
- Support Vector Machines
- Practice: Scikit-learn

Difference between Bagging and Boosting

- it can be used for classification and regression too
- helps to **reduce variance and bias**

What was **bagging**?

creates multiple copies of the original data: constructs several decision trees on the copies and combining all the trees to make predictions

We construct these trees independently!

What is **boosting**?

the decision trees are grown sequentially so each tree is grown using information from previously grown trees $\rightarrow$ boosting is a sequential learning algorithm

These trees are not independent of each other!

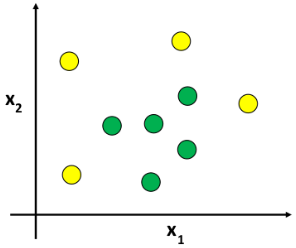
Boosting

A rather counter-intuitive theory: a weak learner is not able to make good predictions

- weak learner is just a bit better than random guess or coin flip For example: decision trees with depth 1
- combining weak learners can prove to be an extremely powerful classifier
- by fitting small trees (decision stumps) we slowly improve the final result in cases when it does not perform well
- We will consider adaptive boosting **"AdaBoost" algorithm**

Boosting: example

What is the task?



- we want to classify the dots

$\rightarrow$ two features

$+$ two output classes (yellow and green)

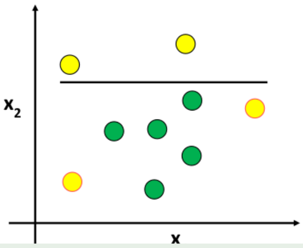
- **Boosting:** combines very

simple weak learners such as decision trees

with depth 1 (capable of linear classification)

Boosting: example

What is the task?



- we want to classify the dots

→ two features

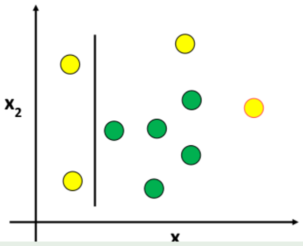
+ two output classes (yellow and green)

- **Boosting:** combines very simple weak learners such as decision trees with depth 1 (capable of linear classification)

- the classifier made 2 mistakes: two yellow dots are misclassified
- → in the next iteration it will focus on the misclassified items
- we'll increase the weights of the misclassified items and decrease the weights of the correctly classified items

Boosting: example

What is the task?



- we want to classify the dots

→ two features

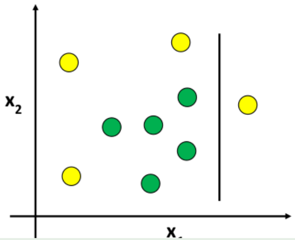
+ two output classes (yellow and green)

- **Boosting:** combines very simple weak learners such as decision trees with depth 1 (capable of linear classification)

- the classifier made 2 mistakes: two yellow dots are misclassified
- → in the next iteration it will focus on the misclassified items
- we'll increase the weights of the misclassified items and decrease the weights of the correctly classified items

Boosting: example

What is the task?



- we want to classify the dots

$\rightarrow$ two features

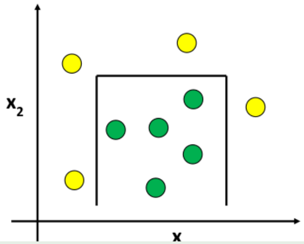
$+$ two output classes (yellow and green)

- **Boosting:** combines very simple weak learners such as decision trees with depth 1 (capable of linear classification)

- the classifier made 3 mistakes: three yellow dots are misclassified
- $\rightarrow$ in the next iteration it will focus on the misclassified items
- we'll increase the weights of the misclassified items and decrease the weights of the correctly classified items

Boosting: example

We just have to combine these weak classifiers!



- we want to classify the dots

$\rightarrow$ two features

$+$ two output classes (yellow and green)

- **Boosting:** combines very simple weak learners such as decision trees with depth 1 (capable of linear classification)

- the classifier made 2 mistakes: two yellow dots are misclassified
- $\rightarrow$ in the next iteration it will focus on the misclassified items
- we'll increase the weights of the misclassified items and decrease the weights of the correctly classified items

Math

- we keep combining $h(x)$ weak learners (each learner knows just a little fraction of the space): $H(x) = sign \sum_{i=1}^{n} \alpha_i h_i(x)$ $\rightarrow$ Final model, not a weak learner anymore
- we assign $+1$ and -1 for the output classes (yellow and green) $\rightarrow$ true class denoted by y
- we initialize the (other) **weight parameters** at the beginning $w_i = 1/N$
- N is the number of the data
- make sure this is a **distribution** $\sum_{i=1}^{N} w_i = 1$
- $\varepsilon = \sum_{wrong} w_i$: the **error** is the sum of the misclassified weights

Math: Algorithm

The algorithm

initialize the weights $\qquad {w_i}^1 = \frac{1}{N}$

$\downarrow$

pick $h_t(x)$ tree that minimizes
the ε_t error term $\qquad \varepsilon_t = \sum_{wrong} {w_i}^t$

$\downarrow$

we can calculate α_t $\qquad \alpha_t = \frac{1}{2} \log \left(\frac{1-\varepsilon_t}{\varepsilon_t} \right)$

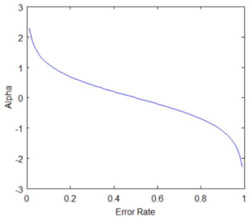
$\downarrow$

update w^{t+1} weights $\qquad {w_i}^{t+1} = \frac{{w_i}^t}{Z} e^{-\alpha_t h_t(x_i) y(x_i)}$
Z: w^{t+1} should be a distribution

On every iteration we add a new $h(x)$ weak learner to the final model

Math: Algorithm

What is $\alpha_t = \frac{1}{2}\log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$? With given $h(x)$ classifier...



- α increases as the error converges to 0:

of course, good classifiers are given more weight

- α value is 0 if the error is 0.5.

Why? Because it is a random guess (coin toss)

$\rightarrow$ we

do not want our algorithm to rely on random guesses

- we give negative α value for $h(x)$ classifiers that are worse than random guesses

  $\Rightarrow$ we do the opposite thats the best action

- This α parameter has something to do with the $h(x)$ learners

Math: Algorithm

What? What is ${w_i}^{t+1} = \frac{{w_i}^t}{Z} e^{-\alpha_t h_t(x_i) y(x_i)}$? $\leftarrow$ weight updating

- the w weights have something to do with the dataset
- we set higher weights to more important samples and lower weight values to less important ones
- the Z makes sure w is distribution so the sum is 1
- $y(x)$ flips the sign of the exponent if $h(x)$ is wrong
  Why is it good? It makes sure to assign smaller weights to samples that are correctly classified and bigger weights for misclassification
  $\rightarrow$ in the next iteration the next $h(x)$ learner can focus on those samples with higher weights
- Why to use α in the formula? This is how we make sure that stronger classfiers' decisions are more important. If a weak classifier misclassifies an input we do not take that as seriously as a strong classifier's mistake

Summary

Bagging	Boosting
every item has the same probability to appear in a new dataset	the samples are weighted so some of them will occur more often
parallel training stage	builds learners in sequential way
final decision is the average of the N learners	final decision is the weighted average of the N learners
reduces variance and solves overfitting	reduces bias but increases over-fitting a bit

XGBoost

$$\begin{cases} \text{Random Forest} \\ \text{Adaboost} \end{cases} (95{-}97) \Longrightarrow \text{Gradient Boosting}(99) \Longrightarrow \begin{cases} \text{Gradient boosting} \\ +\text{regularization} \end{cases}$$

- eXtreme Gradient Boosted trees
- boosting $\subset$ ensemble methods $\Longrightarrow$ Each tree boost attributes that led to misclassifications of previous tree
- **Tabular data**

Advantages

- Routinely wins Kaggle competitions
- Easy to use, Fast
- High Model Performance
- A gooid choice for an algorithm to start with

XGBoost

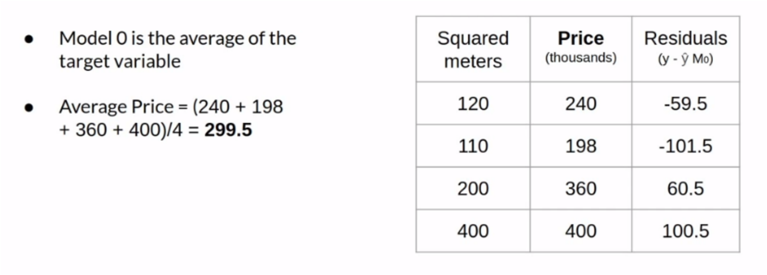
Advantages - Features

- Regularized boosting $\Longrightarrow$ prevents overfitting
- can handle missing values automatically
- parallel processing, GPU
- tree pruning
- cross-validation at each iteration $\Longrightarrow$ enables early stopping
- sparse data

`https://arxiv.org/abs/1603.02754`

Intuition - regression

House price prediction

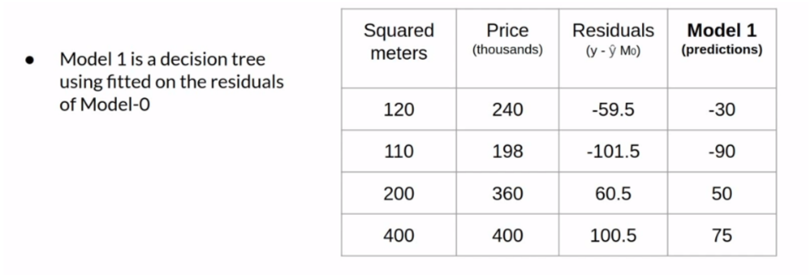


- Model 0 is the average of the target variable
- Average Price = (240 + 198 + 360 + 400)/4 = **299.5**

Squared meters	**Price** (thousands)	Residuals (y - $\hat{y}$ M_0)
120	240	-59.5
110	198	-101.5
200	360	60.5
400	400	100.5

Intuition - regression

House price prediction

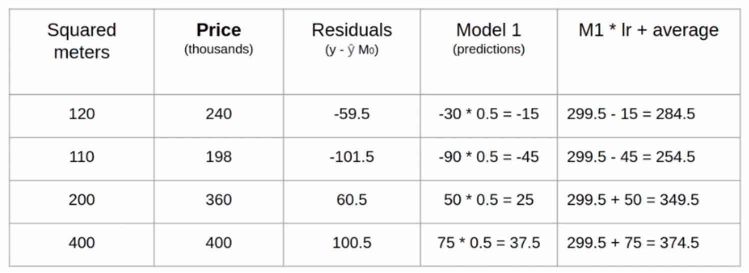


- Model 1 is a decision tree using fitted on the residuals of Model-0

Squared meters	Price (thousands)	Residuals $(y - \hat{y}\ M_0)$	**Model 1 (predictions)**
120	240	-59.5	-30
110	198	-101.5	-90
200	360	60.5	50
400	400	100.5	75

Intiution - regression

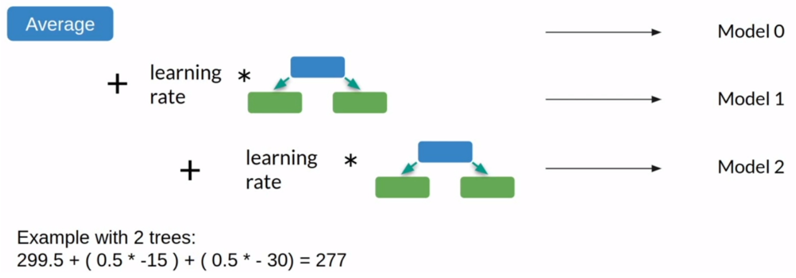
Including the learning rate



Squared meters	**Price** (thousands)	Residuals $(y - \hat{y}\ M_0)$	Model 1 (predictions)	M1 * lr + average
120	240	-59.5	-30 * 0.5 = -15	299.5 - 15 = 284.5
110	198	-101.5	-90 * 0.5 = -45	299.5 - 45 = 254.5
200	360	60.5	50 * 0.5 = 25	299.5 + 50 = 349.5
400	400	100.5	75 * 0.5 = 37.5	299.5 + 75 = 374.5

Intuition - regression

Repeat the process multiple times



Example with 2 trees:
299.5 + (0.5 * -15) + (0.5 * - 30) = 277

Hyperparameters

Critical Hyperparameters

- *learning_rate*: controls how much weight each tree has on the final prediction
- *num_boost_round*: defines the number of trees (number of iterations)
- *max_depth*: controls the depth of the tree
- γ: gamma - minimum require loss for the model to justify a split
- λ: lambda - reguralization on the weights
- *subsample*: percent of training data (rows) used for training a tree at each iteration
- *colsample_bytree*: subsample ratio of columns when constructing each tree

$$\text{Loss} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{t-1} + f_t(x_t)\right) + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2$$

Howto use?

When to use?

- large number of observations ($>$1000)
- less features ($<$100)
- performs well when data has mixture numerical and categorical features or just numeric features

When don't use?

- computer vision,
- nlp

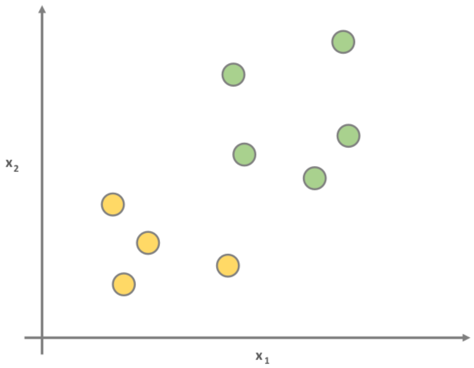
Support Vector Machines

- very popular and widely used supervised learning algorithm (especially for **classification**)
- the great benefit is that it can **operate even in infinite dimensions**
- it defines a **margin** (decision boundary) between the data points in the multidimensional space
- that goal is to find a flat boundary (margin - optimal hyperplane in some sense) that leads to a homogeneous partition of the data
- a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class since in general the larger the margin the lower the generalization error of the classifier

So with Support Vector Machine algorithm we maximize the margin.

Support Vector Machines

What is the aim?



- we

want to find a **hyperplane**

- in this

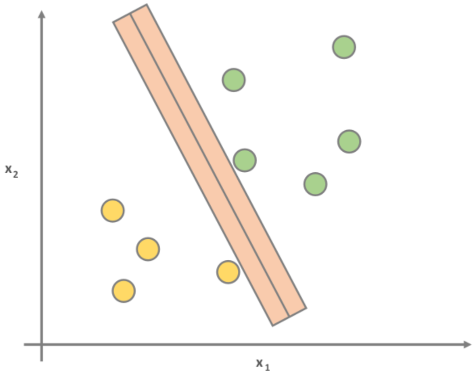
case a line that **separates**

the data points with

the **maximum margin**

Support Vector Machines

What is the aim?



- we

want to find a **hyperplane**

- in this

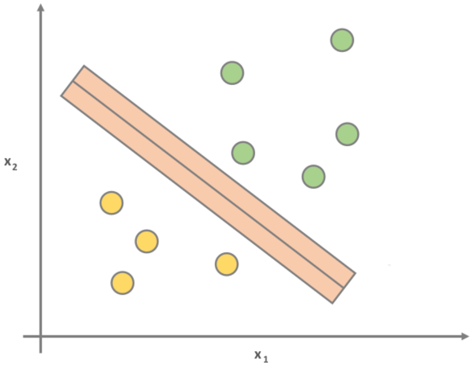
case a line that **separates**

the data points with

the **maximum margin**

Support Vector Machines

What is the aim?



- we

want to find a **hyperplane**

- in this

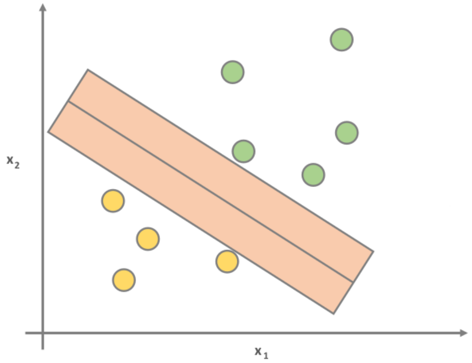
case a line that **separates**

the data points with

the **maximum margin**

Support Vector Machines

The optimal margin



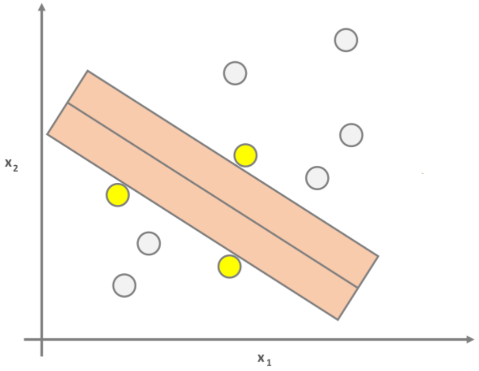
- we

are looking for the margin (decision boundary) with the largest width

This is the maximum margin

Support Vector Machines

The optimal margin: support vectors



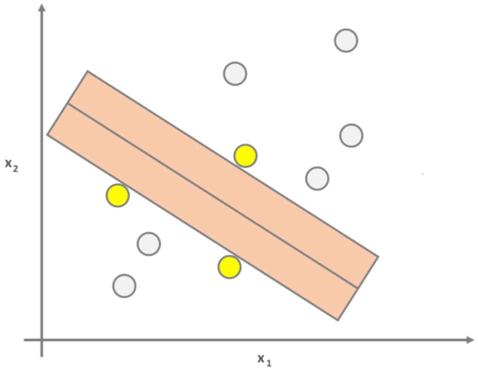
- the points that are closest to the maximum margin are called the **support-vectors**

- storing the information for the decision boundary

- What if we change the other data points? Nothing

Support Vector Machines

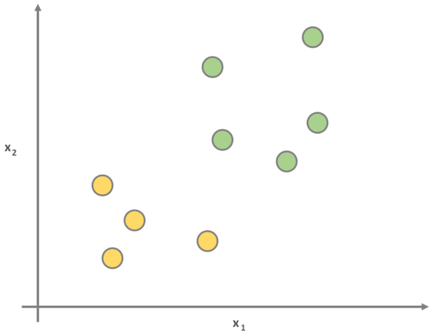
The optimal margin: **support vectors**



- the points that are closest to the maximum margin are called the **support-vectors**
- storing the information for the decision boundary
- but if we change the support vectors ⇒ the model changes as well

Math

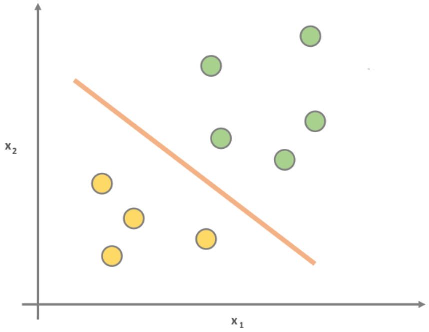
The Hyperplane. What is a hyperplane?



in geometry a hyperplane is
a subspace that has one dimension
fewer than its ambient space
- the hyperplane
separates the space into two parts
- the general equaion
for a hyperplane is $\vec{w}^T\vec{x} + b = 0$

Math

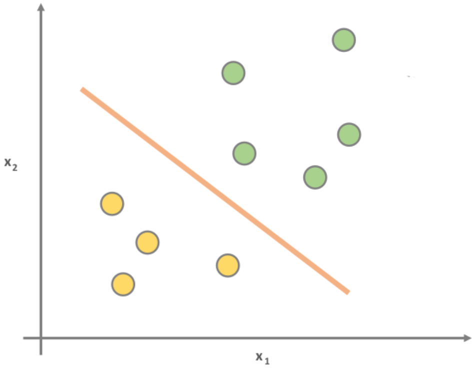
The Hyperplane



in geometry a hyperplane is
a subspace that has one dimension
fewer than its ambient space
- the hyperplane
separates the space into two parts
- the general equaion
for a hyperplane is $\vec{w}^T\vec{x} + b = 0$
- $x_2 = ax_1 + b$

Math

The Hyperplane



in geometry

a hyperplane is a subspace

that has one dimension

fewer than its ambient space

- the hyperplane separates

the space into two parts

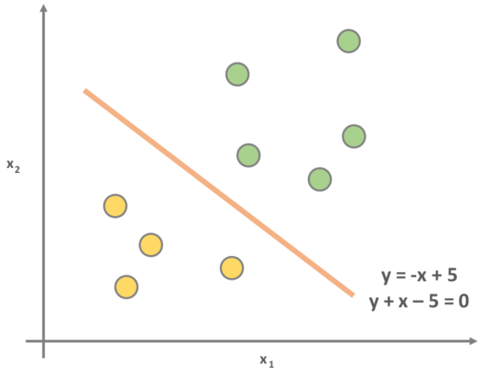
- the general equaion for a

hyperplane is $\vec{w}^T\vec{x} + b = 0$

- $[1, -a] \cdot [y, x]^T - b = 0$

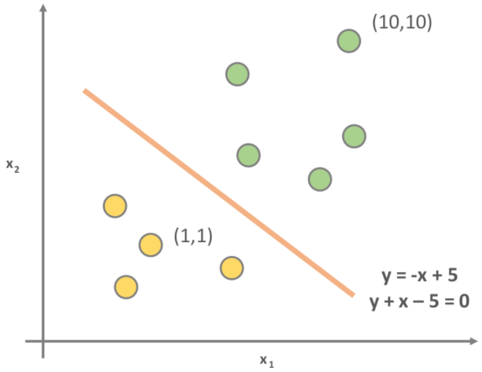
Math: Example

The Hyperplane



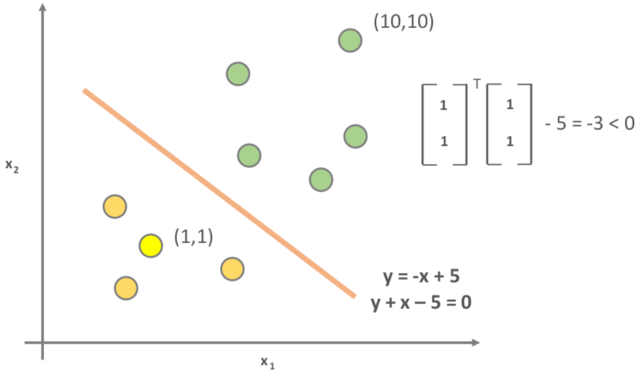
Math: Example

The Hyperplane



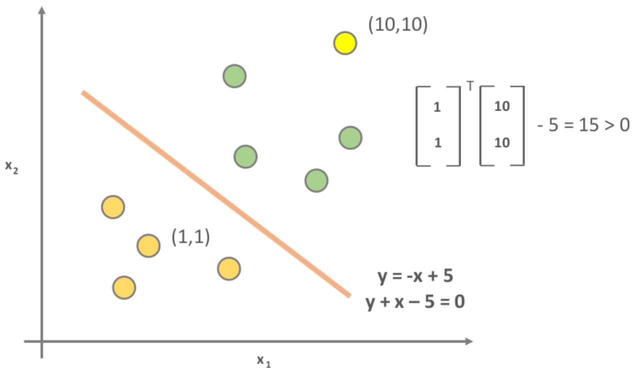
Math: Example

The Hyperplane


(10,10)
(1,1)
x_2
x_1
$\begin{bmatrix}1\\1\end{bmatrix}^T\begin{bmatrix}1\\1\end{bmatrix} - 5 = -3 < 0$
$y = -x + 5$
$y + x - 5 = 0$

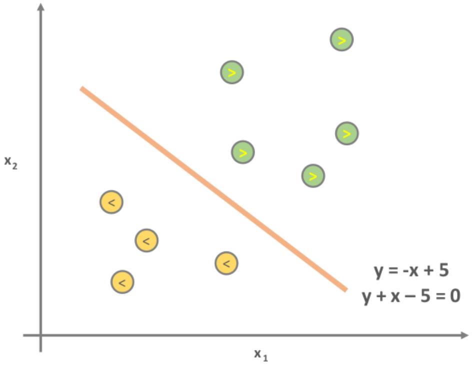
Math: Example

The Hyperplane


(10,10)
(1,1)
$\begin{bmatrix}1\\1\end{bmatrix}^T \begin{bmatrix}10\\10\end{bmatrix} - 5 = 15 > 0$
$y = -x + 5$
$y + x - 5 = 0$
x_2
x_1

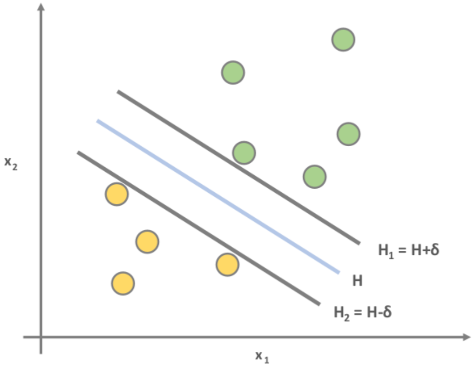
Math: Example

The Hyperplane



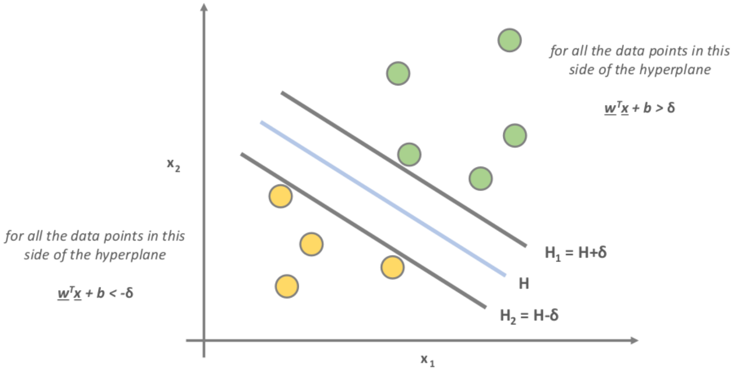
Math: in general

The Hyperplane



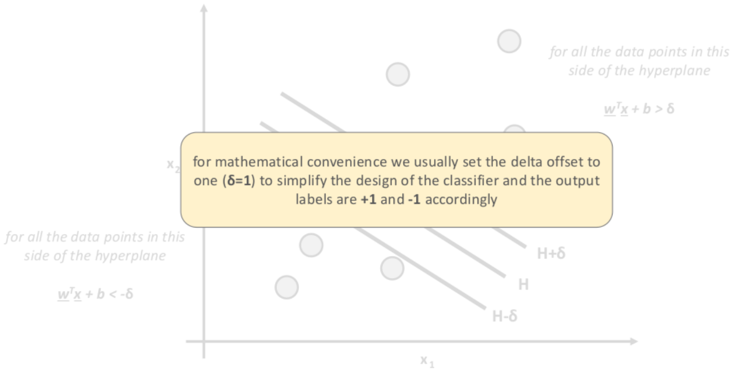
Math: in general

The Hyperplane


for all the data points in this side of the hyperplane
$\underline{w}^T\underline{x} + b > \delta$
x_2
for all the data points in this side of the hyperplane
$\underline{w}^T\underline{x} + b < -\delta$
$H_1 = H+\delta$
H
$H_2 = H-\delta$
x_1

Math: in general

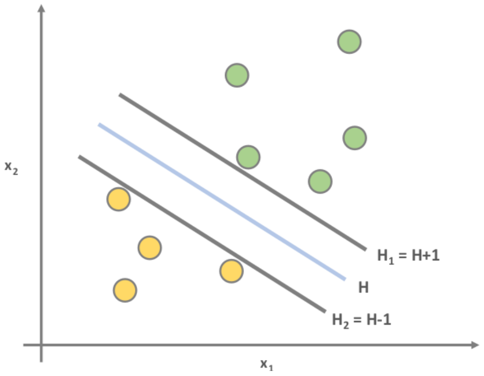
Important assumption


for all the data points in this side of the hyperplane
$\underline{w}^T\underline{x} + b > \delta$
x_2
for all the data points in this side of the hyperplane
$\underline{w}^T\underline{x} + b < -\delta$
H+δ
H
H-δ
x_1

for mathematical convenience we usually set the delta offset to one (**δ=1**) to simplify the design of the classifier and the output labels are **+1** and **-1** accordingly

Math: in general

The constraints in one equation



It is convenient to define $\delta=1$ *because it allows the classification constraints to be* ***combined into a single constraint***

we define the **H** *hyperplane such that*
$\underline{w}\ \underline{x}_i + b \geq 1$ *when* $y_i = 1$ *and*
$\underline{w}\ \underline{x}_i + b \leq -1$ *when* $y_i = -1$

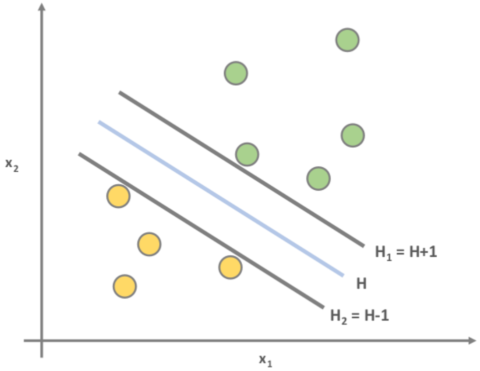
$y_i(\underline{w}\ \underline{x}_i + b) \geq 1$

this is the single constraint we have to deal with !!!

(and of course support vectors are when $y_i(\underline{w}\ \underline{x}_i + b) = 1$*)*

Math: in general

The distance



we define the ***H*** *hyperplane such that*
$\underline{w}\,\underline{x}_i + b \geq 1$ *when* $y_i = 1$ *and*
$\underline{w}\,\underline{x}_i + b \leq -1$ *when* $y_i = -1$

points on the H_1 *and* H_2 *lines are called the* ***support vectors***

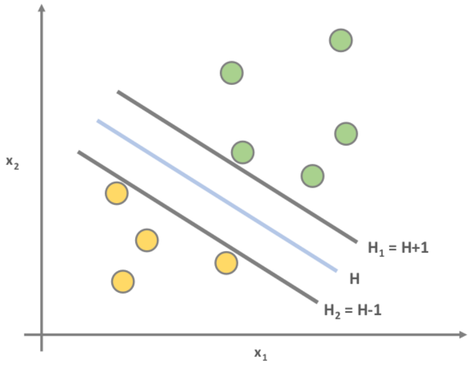
the distance between H_1 *and* H_2 *lines is* $\delta_+ + \delta_-$

WHAT IS THE TOTAL DISTANCE BETWEEN H_1 ***AND*** H_2***?***

it is crucial because ***SVM*** *is a maximum margin classifier so we have to* ***maximize the distance***

Math: in general

Maximization problem



*the distance between **H** and **H_1** (and this is the distance between **H** and **H_2** as well) is defined by the following formula*

$$\frac{1}{||\mathbf{w}||}$$

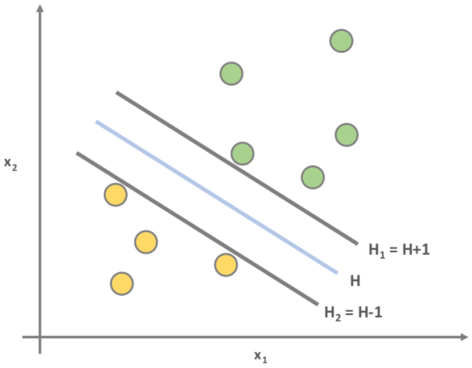
*and if we want to get the total distance between **H_1** and **H_1** then:*

$$\frac{2}{||\mathbf{w}||}$$

*so this is a typical **maximization problem***

Math: in general

Minimization problem



every maximization problem can be ***transformed*** *into a minimization problem*

$$||w||$$

we can ***minimize this function*** *instead and we can transform the problem into:*

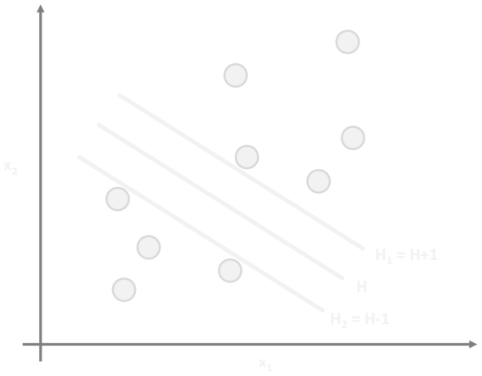
$$\min_{w} \frac{1}{2} ||w||^2$$

$$y_i(\underline{w}\,\underline{x}_i + b) \geq 1 \quad i=1...N$$

so this is a typical ***minimization problem***

Math: in general

Summary



*every maximization problem can be **transformed** into a minimization problem*

$||w||$

PRIMAL FORM OF THE PROBLEM !!!

we can minimize this function instead and we can transform the problem into:

$$\min_{w} \frac{1}{2} ||w||^2$$

$$y_i(\underline{w}\,\underline{x}_i + b) \geq 1 \quad i=1\ldots N$$

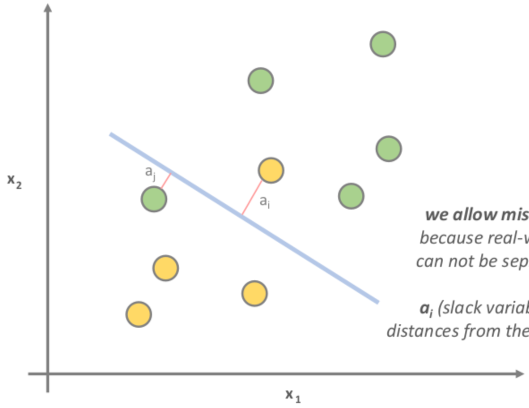
*so this is a typical **minimization problem***

Problem

- the main problem is that in real-world problems are non-linearly separable
- instead of classifying all the data points correctly - we **allow some mistakes**
- in many real-world applications the relationships between variables are usually non-linear
- a key feature of SVMs is their ability to map the problem into a higher dimensional space using a process known as the "kernel trick"
- $\Rightarrow$ non-linear relationship may suddenly appears to be quite linear

Allowing Misclassification

Misclassifications



we allow misclassifications *because real-world problems can not be sepratared linearly*

a_i *(slack variables) define the distances from the decision boundary*

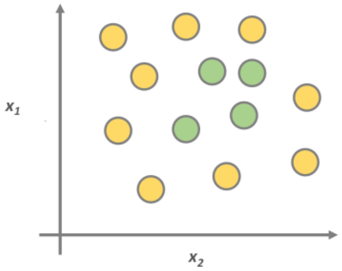
Reformulation

The problem statement

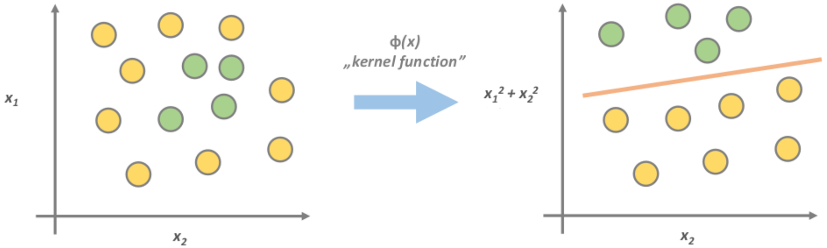
$$\begin{cases} \min_w \left[\frac{1}{2}\|\vec{w}\|^2 + C\sum_i a_i\right] \\ y_i\left(\vec{w}\vec{x}_i + b\right) \geq 1 - a_i, \quad i = 1, \ldots, N \end{cases}$$

- C is the **Cost parameter** to all points that violate the constraints
- a_i is 0 for the points that are classified correctly
- we can tune the C parameter - tuning the **Penalty term** for the data points that are misclassified
- if C is large then the algorithm tries to find a 100% separation
- if C is low then wider overall margin is allowed with more misclassified data points

The kernel trick

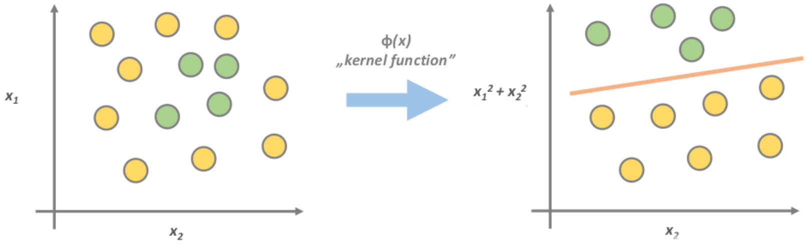


The kernel trick



*with the **kernel function** we can transform the problem into a higher dimensional space that is a linearly separable one (additional variabe is altitude)*

The kernel trick



SVM LEARNS CONCEPTS (FEATURE) THAT WERE NOT EXPLICITLY MEASURED IN THE ORIGINAL DATASET !!!

Summary

- so with the help of the $\Phi(x)$ kernel-function we **transfrom all the points** in the dataset one by one
- and we end up with a **higher dimensional space**
- $K(\vec{x_i}, \vec{x_j}) = \Phi(\vec{x_i})\Phi(\vec{x_j})$

Other Kernels

- linear kernel: $K(\vec{x_i}, \vec{x_j}) = \vec{x_i} \cdot \vec{x_j}$, it does not transform the data
- **Polynomial** kernel: $K(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j} + 1)^d$
- **Gaussian radial basis function** kernel: $K(\vec{x_i}, \vec{x_j}) = \exp\left(-\frac{\|\vec{x_i} - \vec{x_j}\|^2}{2\sigma^2}\right)$

Pros and contras

Advantages	**Disadvantages**
- can be used both regression and classification as well	- it deals with a large number of parameters and kernels
memory friendly	quite slow especially when there is a large number of features
it works fine even in infinite dimensions	there are no probabilites in the predictions