Supervised learning

Inteligencia Artificial en los Sistemas de Control Autónomo
Máster en Ciencia y Tecnología desde el Espacio

Departamento de Automática

Objectives

1. Extend supervised learning algorithms
2. Apply supervised learning to real-world problems

Bibliography

- Müller, Andreas C., Guido, Sarah. *Introduction to Machine Learning with Python.* O'Reilly. 2016

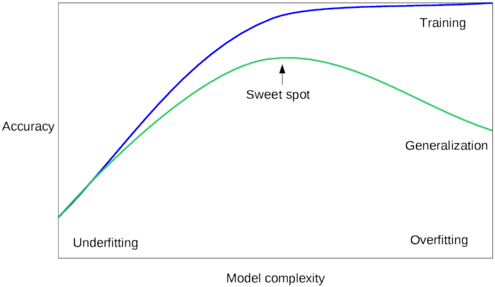
All figures have been taken from `https://github.com/amueller/introduction_to_ml_with_python/blob/master/02-supervised-learning.ipynb`

Table of Contents

Generalization, overfitting and underfitting

Generalization: accurate predictions on unseen data

- i.e. there is no overfitting neither underfitting
- Depends on model complexity and data variability



(Source)

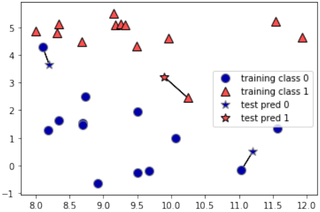
k-Nearest Neighbors

k-NN classification (I)

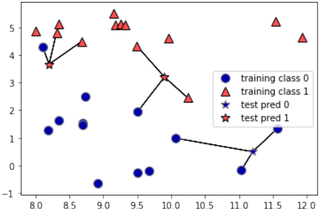
k-NN (k-Nearest Neighbors): Likely, the simplest classifier

- Given a data point, it takes its *k* closests neighbors
- Same prediction than its neighbors

1-NN



3-NN



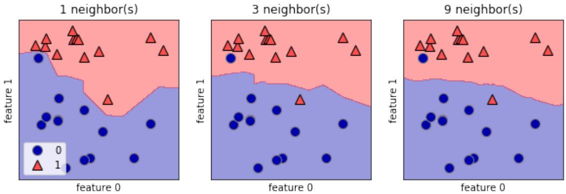
k-NN does not generate a model

- The whole dataset must be stored

k uses to be an odd number (1-NN, 3-NN, 5-NN, ...)

k-Nearest Neighbors

k-NN classification (II)



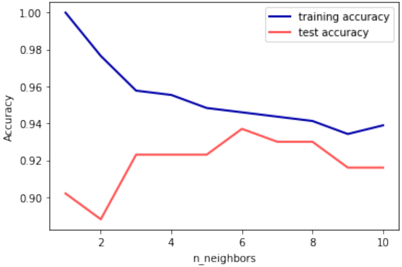
k determines the model complexity

- Smoother boundaries in larger k values
- Model complexity decreases with k
- If k equals the number of samples, k-NN always predicts the most frequent class

How to figure out the best k?

k-Nearest Neighbors

k-NN classification (III)



k-Nearest Neighbors

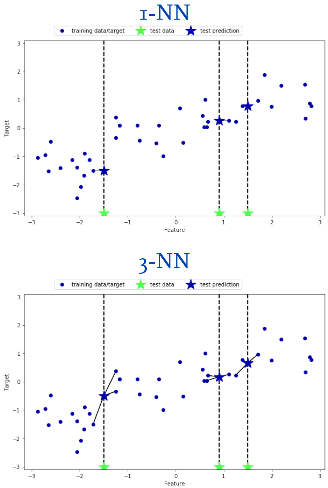
kNN regression (I)

k-NN regression

Given a data point

1. Take the *k* closest data points
2. Predict same target value (1-NN) or averate target value (k-NN)

Performace is measured with a regression metric, by default, R^2

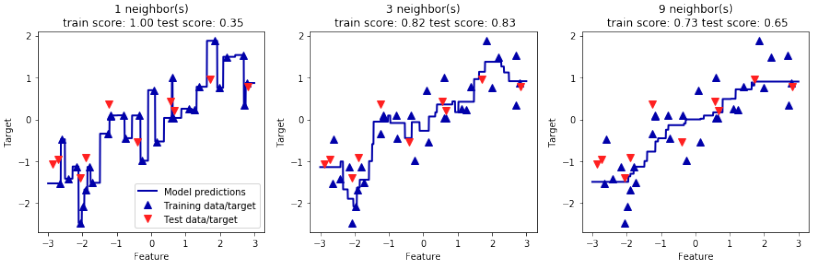


1-NN

3-NN

k-Nearest Neighbors

kNN regression (II)



k determines boundary smoothness

1. With $k = 1$, prediction visits all data points
2. With large k values, fit is worse

k-Nearest Neighbors

Summary

Hyperparameters	Advantages	Disadvantages
k	Simple	Slow with large datasets
Distance	Baseline	Bad performance with hundreds or more attributes
		No model
		Dataset must be stored in memory

Linear models

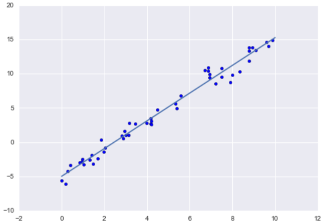
Linear model (I)

Linear model

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

for a single feature $y = \beta_0 + \beta_1 x_1$, where

- β_0 is the intercept
- β_1 is the slope
- Intepretable model



Lineal models assume a linear relationship among variables

- This limitation can be easely overcomed
- Surprisingly good results in high dimensional spaces

Linear models

Linear regression

Different linear models for regression

- The difference lies in how β_i parameters are learned

Ordinary Least Squares (OLS): Minimizes mean squared error

- OLS does not have any hyperparameter
- No complexity control

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2$$

Linear regression can be used to fit non-linear models

- Just adding new attributes

Linear models

Regularized linear models

Regularization: Term that penalizes complexity

- Added to the cost function
- Lineal models remain the same
- Train to minimize cost function and coefficients
- Intercepts are not part of regularization

Three regularizations

- L1 (Lasso regression), L2 (Ridge regression) and ElasticNet (L1 and L2)

Lasso (L1)

$\alpha \sum_j^n |\beta_j|$

Ridge (L2)

$\frac{\alpha}{2} \sum_j^n \beta_j^2$

ElasticNet

$\alpha \left(\frac{\lambda}{2} \sum_j^n \beta_j^2 + (1 - \lambda) \sum_j^n |\beta_j| \right)$

Linear models

Ridge regression

Ridge regression (or L2 regularization) adds a new term to cost function

$$\text{MSE} + \alpha \sum_{i=1}^{n} \beta_i^2$$

α controls the model complexity

- If $\alpha = 0$ Ridge becomes a regular linear regression
- Optimal α depends on the problem

Ridge by default

Linear models

Lasso regression (I)

Lasso regression (or L1 regularization) adds a new term to cost function

$$\text{MSE} + \alpha \frac{1}{2} \sum_{i=1}^{n} |\beta_i|$$

α controls the model complexity

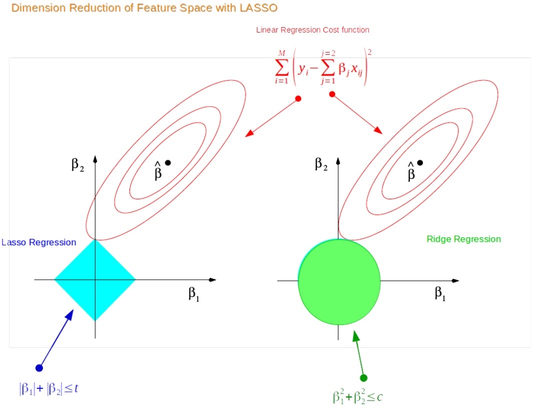
- If $\alpha = 0$ Ridge becomes a regular linear regression
- Optimal α depends on the problem

Some coefficiets may be exactly zero

- Implicit feature selection
- Easier interpretation
- Better with large number of attributes

Linear models

Lasso regression (II)


Dimension Reduction of Feature Space with LASSO
Linear Regression Cost function
$\sum_{i=1}^{M}\left(y_i - \sum_{j=1}^{j=2}\beta_j x_{ij}\right)^2$
β_2
$\hat{\beta}$
Lasso Regression
β_1
β_2
$\hat{\beta}$
Ridge Regression
β_1
$|\beta_1| + |\beta_2| \leq t$
$\beta_1^2 + \beta_2^2 \leq c$

(Source)

Linear models

ElasticNet

Lasso and Ridge can be combined

$$\text{MSE} + \alpha \left(\lambda \frac{1}{2} \sum_{i=1}^{n} |\beta_i| + (1-\lambda) \sum_{i=1}^{n} \beta_i^2 \right)$$

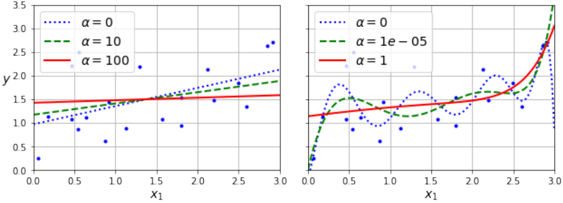
Two hyperparameters

- α controls the model complexity
- λ balances between L1 and L2

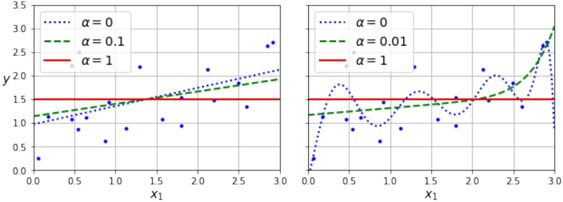
Linear models

Regularized linear models comparison

Ridge - L2



Lasso - L1



Linear models

Linear models for classification (I)

A linear regression can be used as classifier

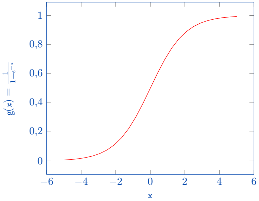
- Just compare the prediction with a threshold (0, for instance)
  - If $\hat{y} > 0$, assign class 1
  - If $\hat{y} <= 0$, assign class -1
- The decision boundary for any binary linal classifier is a line, plane or hyperplane

A logistic regression is a generalization of a linear regression

- It is a binary classifier
- Its output is a probability

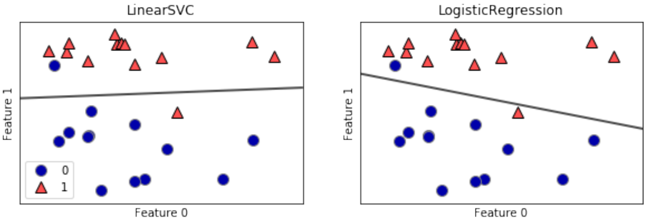
$$p = \sigma\left(\beta_0 + \sum_{i=1}^{n} \beta_i x_i\right),$$

where $\sigma(t)$ is the logistic function, defined as $\sigma(t) = \frac{1}{1+e^t}$



Linear models

Linear models for classification (II)

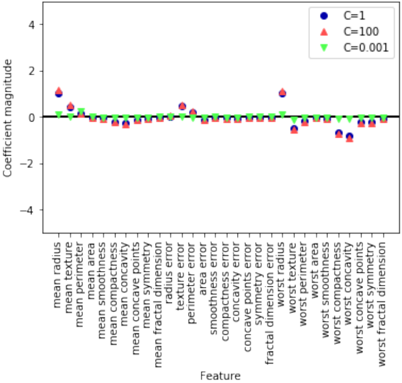


Linear models

Linear models for classification (III)

The model can be regularized with L1, L2 and ElasticNet

- In Scikit-Learn, regularization strength is given by C
- Lower values of C correspond to smaller regularization strength



Linear models

Summary

Hyperparameters	Advantages	Disadvantages
-	Fast train and predict	No complexity tuning
α (L1, L2, ElasticNet)	Scales well to large datasets	Limited in low dimensional spaces
l1_ratio (ElasticNet)	Better in high dimensional spaces	
	Few hyperparameters	
	Interpretable	

Better when the number of features is large compared to the number of samples

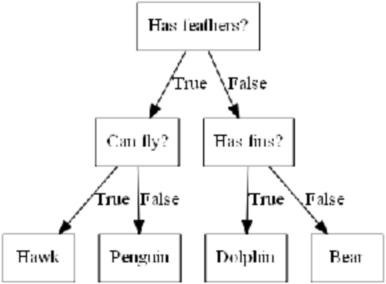
Decision Trees

Decision trees are a family of algorithms for classification and regression

- They learn a tree data strucure
- Hierarchy of if/else questions (test, or node)
- Decision (terminal node or leaf)

Usually, datasets does not contain binary attributes

- Continous features
- Is feature i larger than value *a*?

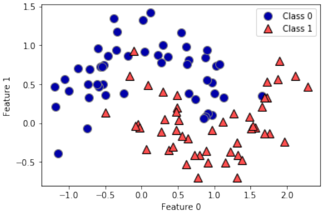


Decision Trees

Building decision trees (I)

Tree learning algorithm

1. Begin with the root node
2. Searches all possible tests (according to a purity measure)
3. The most informative test is taken
4. Repeat recursively

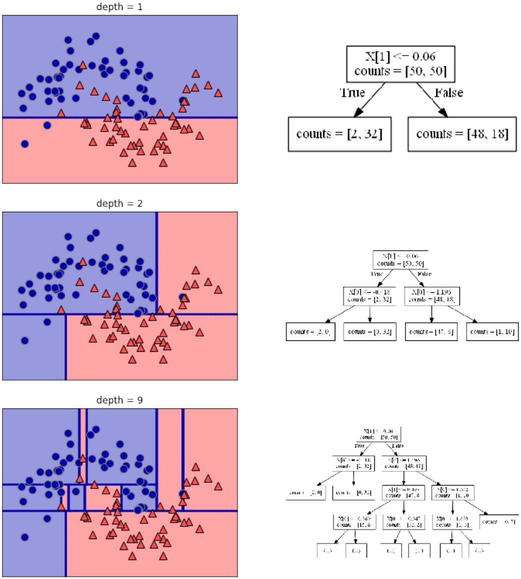


Prediction of a new data point

- Classification: Majority class in the partition
- Regression: Average value of target values in the partition

Decision Trees

Building decision trees (II)



Decision Trees

Building decision trees (III)

Let p_{mk} be the propotion of class k in node m, and Q_m the data in node m

Gini

$$G(Q_m) = \sum_k p_{mk}(1 - p_{ml})$$

Log Loss or Entropy

$$H(Q_m) = -\sum_k p_{mk} \log(p_{ml})$$

Decision Trees

Controlling complexity of decision trees

Trees tend to grow until all leaves are pure

- Very big trees in real problems
- Big trees use to be overfitted models

Two strategies to prevent overfitting

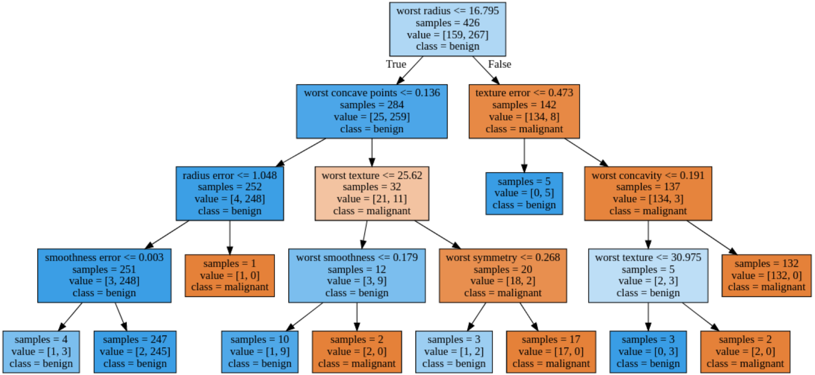
- Pre-prunning: Stop the creation of the tree early accorgind to some criteria
  - Maximum depth, number of leaves, minimum number of points in a node, ...
  - Implemented in Sciki-Learn
- Post-prunning: Build the tree and then remove nodes with little information

Decision Trees

Analyzing decision trees

Decision trees is easily explained to nonexperts

- Interpretable models
- Deep trees are overwhelming
- Trick: Observe the path with most data

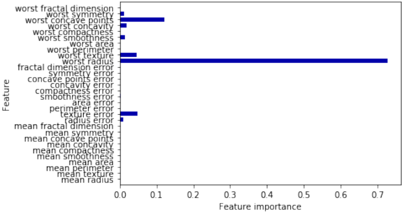


Decision Trees

Analyzing decision trees

Feature importace is a metric that summarizes features

- Number between 0 (not used at all) and 1 (perfect prediction)
- Feature importances sum to one
- Useful for feature selection and model interpretation



Some considerations

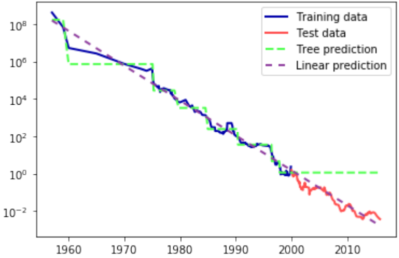
- It does not inform about the relationship between attribute and target
- It quantifies the importance in the tree
  - Correlated attributes may score low importance

Decision Trees

Decision trees in regression

Decision trees are not able to extrapolate

- i. e. to predict outside of the range of the training data
- It is specially important in regression problems



Decision Trees

Summary

Hyperparameters	Advantages	Disadvantages
max_depth	Visualization	Tend to overfit
max_leaf_nodes	Interpretable by non-experts	Poor generalization
min_samples_leaf	Invariant to scale	
criterion	Mix of categorial and numerical data	

Ensembles of Decision Trees

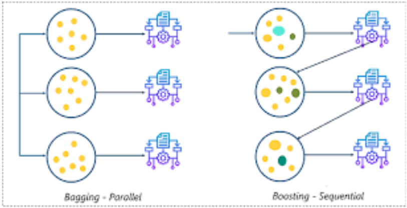
Ensembles

Ensembles, in ML, refers to the combination of several models

- For instance, an ensemble of three classifers voting

Two common aprochaes to build ensembles

- Bagging (or *bootstrap*) samples the dataset with replacement
  - The ensemble make prediction by aggregating its predictors
- Boosting trains models to correct previous models



(Source)

Ensembles of Decision Trees

Random forests

Trees have poor generalization

- A tree is good doing his job, but does not generalize well
- Different trees could overfit in differentways
- Idea: Using many trees and averaging their result

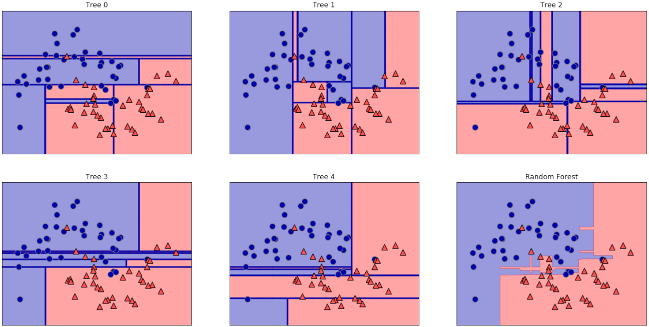
Random forest is an algorithm that trains different trees injecting randomness

- Selecting data - bootstrap
- Selecting features in each test
  - It does not look for the best test
  - It looks for the best test involving a random subset of features
  - The size of the features subset is a critical hyperparameter

Ensembles of Decision Trees

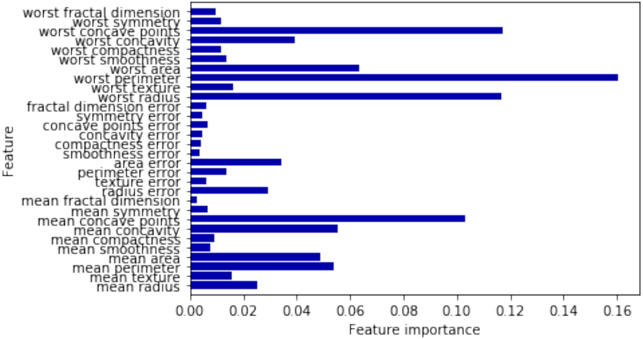
Analyzing random forests (I)

Random forest with five trees



Ensembles of Decision Trees

Analyzing random forests (II)



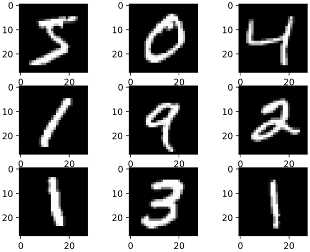
Feature importance can be aggregated

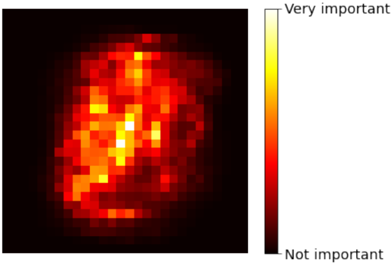
- More informative than single trees
- The algorithms must consider many possible explanations

Ensembles of Decision Trees

Analyzing random forests (III)

Random forest classifier with MNIST dataset





Ensembles of Decision Trees

Summary: Random forest

Hyperparameters	Advantages	Disadvantages
Same than trees	Same than trees	Interpretation
Number of trees	High performance	High dimensional data
Pre-pruning	Robust	Sparse data
	Widely used	Memory and CPU
	Parallelized	

Ensembles of Decision Trees

Gradient boosted regression trees (I)

Gradient boosting is an esemble of classification and regression trees

- Based on boosting, builds trees in a serial manner
- One tree corrects the mistakes of the previous one

A set of *weak learners* is used

- Shallow trees (by default, 3 in Sklearn)
- No data randomization
- Strong pre-pruning

A new hyperparameter: learning rate

- How strongly each tree tries to correct
- High learning rate makes stronger corrections
    - More complex models
- More trees also adds more complexity

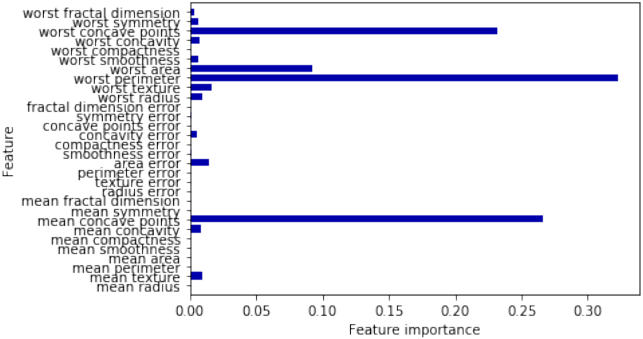
State of the art results

- Widely adopted by industry
- Comparable in performance with deep neural networks

Ensembles of Decision Trees

Gradient boosted regression trees (II)

Feature importances with cancer dataset



The (XGBoost) package provides a high performance implementation of gradient boosted trees

Ensembles of Decision Trees

Summary

Hyperparameters	Advantages	Disadvantages
Same than trees	Very high performance	Slow
Number of trees	Invariant to scale	High dimensional data
Learning rate	Mix of categorial and numerical data	Tricky hyperparameter tuning
		Overfitting

Support Vector Machines

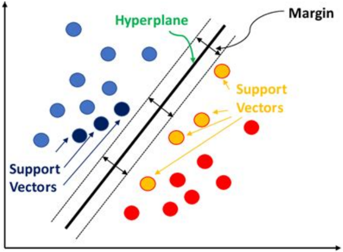
Linear SVM (I)

SVM, or *Support Vector Machines*, is a popular and flexible learning model

- Classification, regression and outlayer detection
- Linear and non-linear models
- Quite popular with small and medium datasets

Learning algorithm

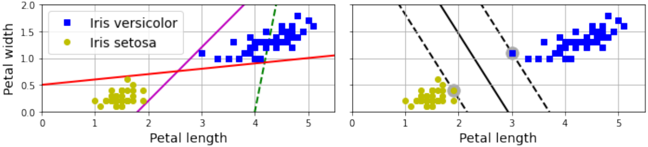
1. It localizes data points in the boundary of the classes
    - They are named *support vectors*
2. Determine an hyperplane that splits them maxizing *margin*



(Source)

Support Vector Machines

Linear SVM (II)

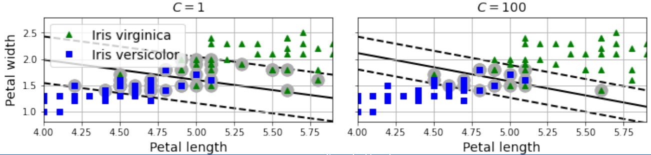


Two big problems with hard margins

- Most datasets are not linearly separable
- Outlayers

We look for a balance between good fit and margin violations: C

- C sets the tolerance to margin violations
- Low C → Low tolerance

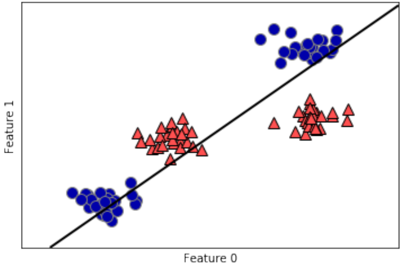


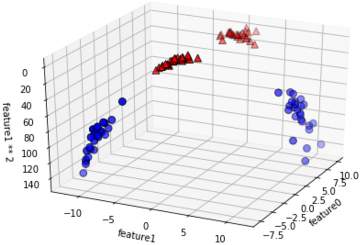
Support Vector Machines

Linear models and nonlinear features (I)

Plain SVMs are limited in low-dimensional spaces

- Lines, planes and hyperplanes
- Adding new features is a way to overcome this limitation

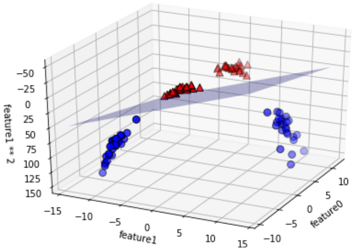


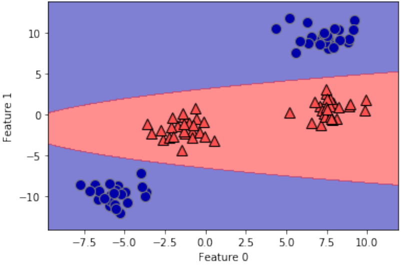


$$feature\_3 = feature\_1^2$$

Support Vector Machines

Linear models and nonlinear features (II)





Support Vector Machines

The kernel trick

Adding nonlinear attributes makes linear models much more powerful

- Which features should we add?
- How we compute interations in a 100-dimensional feature space?

Some mathematical magic: The *kernel trick*

- It computes data distances for expanded feature representation ...
- ... without computing the expansion!

It applies a function named `kernel`

- Polynomial kernel, up to a certain degree
- Radial basis function (RBF) kernel (Gaussian kernel)
- Linear kernel, no expansion is done

The kernel trick can be used in other techniques like PCA

Support Vector Machines

Understanding SVMs (I)

To predict a new point, the distance to each of the support vector is computed

- Distance is measured by the Gaussian kernel
- Decision is taken based on the distance and learned importance

$$k_{rbf}(x_1, x_2) = \exp(-\gamma||x_1 - x_2||^2)$$

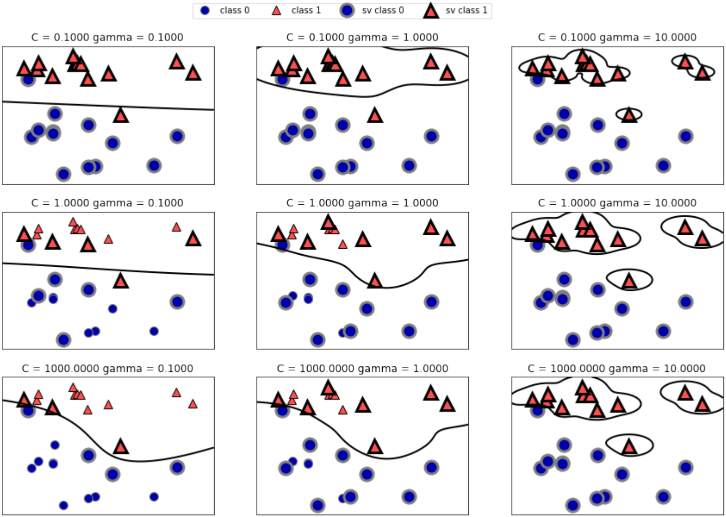
where $||\dot{|}|$ denotes Euclidean distance and γ is an hyperparameter

- γ determines how far the influence of a single point reaches
- Low γ, higher complexity

Remember, C is a regularization parameter

Support Vector Machines

Understanding SVMs (II)

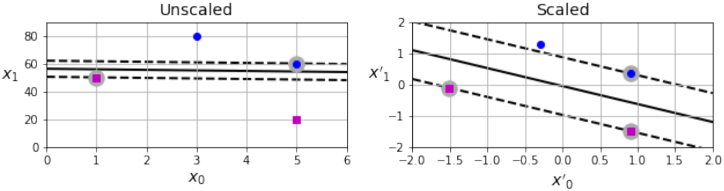


Support Vector Machines

Understanding SVMs (III)

SVM is very sensitive to scale

- Always use standarized or normalized data



Support Vector Machines

Summary

Hyperparameters	Advantages	Disadvantages
C	Powerful	Memory and CPU
γ	Low and high dimensional	Number of samples
Kernel	Flexible	Scaling
		No interpretable