Supervised learning

Aprendizaje Automático para la Robótica
Máster Universitario en Ingeniería Industrial

Departamento de Automática

Universidad de Alcalá ISG

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

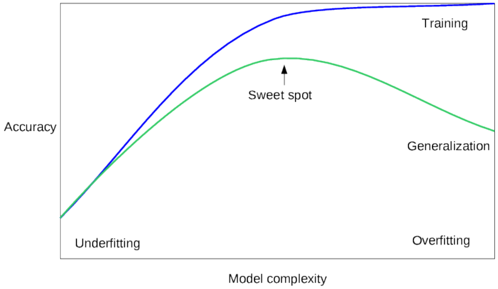
All figures have been taken from `https://github.com/amueller/introduction_to_ml_with_python/blob/master/02-supervised-learning.ipynb`

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)

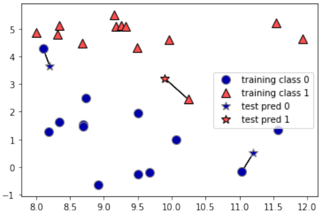
k-Nearest Neighbors

k-NN classification (I)

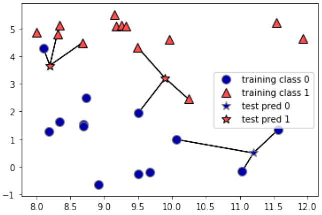
k-NN (k-Nearest Neighbors): Likely, the simplest classifier

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

1-NN



3-NN



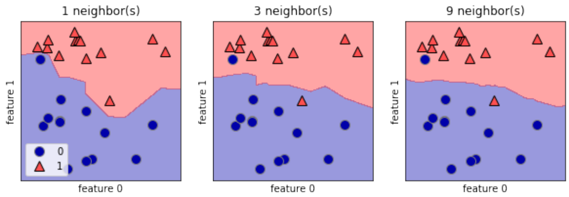
k-NN does not generate a model

- 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)



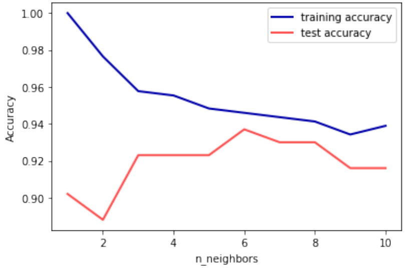
k determines the model complexity

- 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 classifier

Scikit-learn

`sklearn.neighbors.KNeighborsClassifier`

Constructor arguments:

- `n_neighbors`: int, default=5
- `metric`: string, default='minkowski'
- `p`: int, default=2 ($p = 1$ Manhatan distance, $p = 2$ euclidean distance)

Attributes:

- `classes_`: ndarray (n_samples)

Methods: `fit()`, `predict()`

(Scikit-Learn reference)

k-Nearest Neighbors

kNN regression (I)

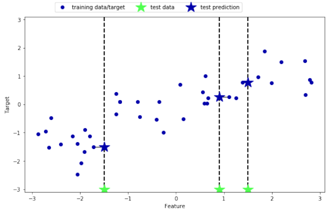
k-NN regression

Given a data point

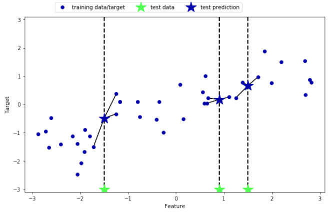
1. Take the *k* closest data points
2. Predict same target value (1-NN) or averate target value (k-NN)

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

1-NN

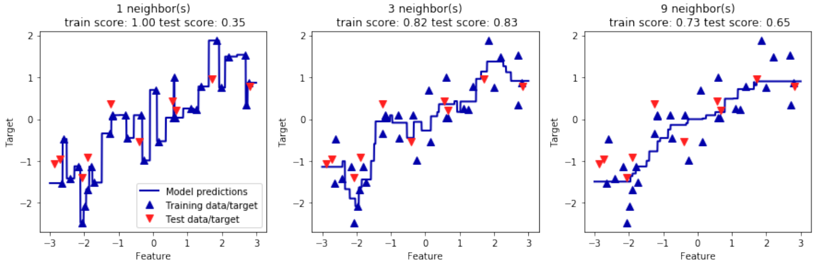


3-NN



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 regressor

Scikit-learn

`sklearn.neighbors.KNeighborsRegressor`

Constructor arguments:

- `n_neighbors`: int, default=5
- `metric`: string, default='minkowski'
- `p`: int, default=2 ($p = 1$ Manhatan distance, $p = 2$ euclidean distance)

Attributes:

Methods: `fit()`, `predict()`

(Scikit-Learn reference)

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

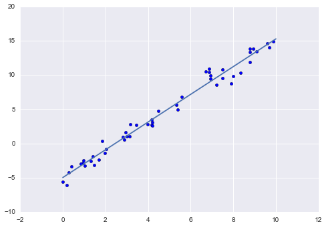
Linear model (I)

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

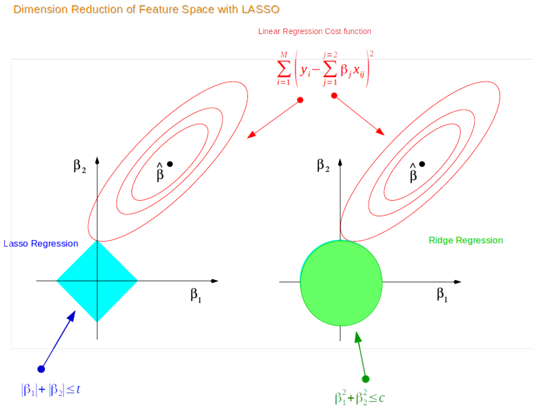
- If $\alpha = 0$ Ridge becomes a regular linear regression
- Optimal α depends on the problem

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
$\hat{\beta}$
β_1
Lasso Regression
Ridge Regression
$|\beta_1|+|\beta_2|\leq t$
$\beta_1^2+\beta_2^2\leq c$

(Source)

Linear models

ElasticNet

Lasso and Ridge can be combined

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

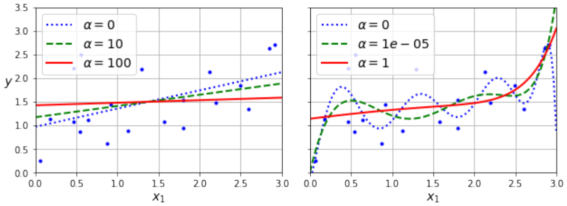
Two hyperparameters

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

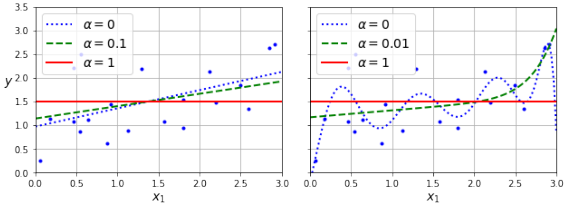
Linear models

Regularized linear models comparison

Ridge - L2



Lasso - L1



Linear models

Linear models for classification

Three regularizations

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

Lasso

$\lambda \sum_j^n \beta_j^2$

Linear models

Scikit-learn

TODO

`sklearn.cluster.AgglomerativeClustering`

Constructor arguments:

- `linkage`: 'ward', 'complete', 'average', 'single'

Attributes:

- `n_clusters`: int
- `labels_`: ndarray (n_samples)

Methods: `fit()`, `fit_predict()`

(Scikit-Learn reference)

Linear models

Summary (I)

Linear regression

Hyperparameters	Advantages	Disadvantages
-	Fast train and predict Scales well to large data-sets	No complexity tuning

Ridge regression

Hyperparameters	Advantages	Disadvantages
α	Election by default	

Linear models

Summary (II)

Lasso regression

Hyperparameters	Advantages	Disadvantages
α	Interpretation	

ElasticNet

Hyperparameters	Advantages	Disadvantages
α		
λ		

Naive Bayes Classifiers

TODO

Naive Bayes Classifiers

Scikit-learn

`sklearn.cluster.AgglomerativeClustering`

Constructor arguments:

- `linkage`: 'ward', 'complete', 'average', 'single'

Attributes:

- `n_clusters`: int
- `labels_`: ndarray (n_samples)

Methods: `fit()`, `fit_predict()`

(Scikit-Learn reference)

Naive Bayes Classifiers

Summary

Hyperparameters	Advantages	Disadvantages

Decission Trees

TODO

Decission Trees

Scikit-learn

sklearn.cluster.AgglomerativeClustering

Constructor arguments:

- `linkage`: 'ward', 'complete', 'average', 'single'

Methods: `fit()`, `fit_predict()`

Attributes:

- `n_clusters`: int
- `labels_`: ndarray (n_samples)

(Scikit-Learn reference)

Decission Trees

Summary

Hyperparameters	Advantages	Disadvantages

Ensembles of Decision Trees

TODO

Ensembles of Decision Trees

Ensembles of Decision Trees : Scikit-learn

`sklearn.cluster.AgglomerativeClustering`

Constructor arguments:

- `linkage`: 'ward', 'complete', 'average', 'single'

Attributes:

- `n_clusters`: int
- `labels_`: ndarray (n_samples)

Methods: `fit()`, `fit_predict()`

(Scikit-Learn reference)

Ensembles of Decision Trees

Summary

Hyperparameters	Advantages	Disadvantages

Support Vector Machines

TODO

Support Vector Machines

Kernelized Support Vector Machines

TODO

Scikit-Learn

Support Vector Machines

Scikit-learn

`sklearn.cluster.AgglomerativeClustering`

Constructor arguments:

- `linkage`: 'ward', 'complete', 'average', 'single'

Attributes:

- `n_clusters`: int
- `labels_`: ndarray (n_samples)

Methods: `fit()`, `fit_predict()`

(Scikit-Learn reference)

Support Vector Machines

Summary

Hyperparameters	Advantages	Disadvantages

A

B

TODO

A

B: Scikit-learn

`sklearn.cluster.AgglomerativeClustering`

Constructor arguments:

- `linkage`: 'ward', 'complete', 'average', 'single'

Attributes:

- `n_clusters`: int
- `labels_`: ndarray (n_samples)

Methods: `fit()`, `fit_predict()`

(Scikit-Learn reference)

A

B: Summary

Hyperparameters	Advantages	Disadvantages

Algorithms

ARIMA (I)

AR: Autoregressive model

- Current observation depends on the last p observations
- Long term memory

AR(p)

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-1} + \epsilon_t$$

MA: Moving Average model

- Current observation linearly depends on the last q innovations
- Short term memory

MA(q)

$$X_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + \ldots + \theta_q \epsilon_{t-q}$$

ARMA model = AR + MA

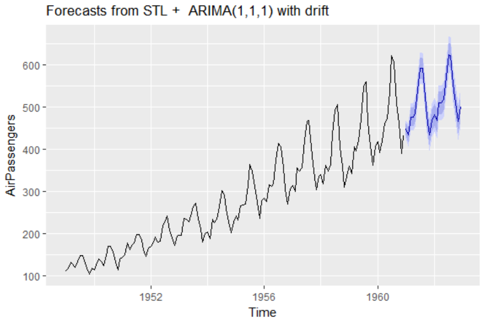
- ARMA(p, q): Two hyperparameters, p and q

Algorithms

ARIMA (II)

ARIMA = AR + i + MA (AR integrated MA)

- ARIMA(p, d, q)
- Three integer parameters: p, q and d (in practice, low order models)



(Source)

autoarima: search over p, q and d