TECHNISCHE UNIVERSITÄT DRESDEN

SEMANTIC COMPUTING

Lecture 5: Supervised Machine Learning

Dagmar Gromann

International Center For Computational Logic

TU Dresden, 16 November 2018

Overview

- Linear Regression
- Support Vector Machines (SVM)
- Decision Trees

Supervised Machine Learning algorithms

- Naive Bayes for classification problems
- Linear regression for regression problems
- Support vector machines for classification and regression problems
- Decision trees for classification and regression problems
- Random forest for classification and regression problems

Linear Regression

Linear Regression

Definition

The goal of linear regression is to predict the value of one or more continuous target variables $\hat{y}$ given the value of a d dimensional vector of input variables x.

- Reasonable if strong linear relationship in data or linear model is a good approximation: $h(x) = w_0 + w_1 x_1 + ... + w_n x_n$ where $w_0, ..., w_n$ are the coefficients to be found/minimized and $x_1, ..., x_n$ are the input features
- fitting coefficients with training data based on the **error or loss function**, e.g. sum of squared errors

$$L = \frac{1}{2}\sum_{i=1}^{m}(\hat{y} - y)^2 \quad \#\hat{y} = prediction = output\ of\ h(x)$$

Regression Example: Data

A diabetes dataset describes 10 physiological variables (see below) measured on 442 patients and an indication of the progression of the disease after one year.

```
from sklearn import datasets
diabetes = datasets.load_diabetes()
print(diabetes.feature_names)
```

age	sex	Body mass index	average blood pressure	y
59	2	32.1	101	252
19	1	19.2	87	137
48	1	21.6	87	75

The task is to predict disease progression from physiological variables.

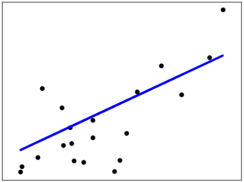
Source original dataset: `https://www4.stat.ncsu.edu/~boos/var.select/diabetes.tab.txt`

Regression Example: Code and Output (1/2)

```
from sklearn import linear_model

diabetes_X_train = diabetes.data[:-20]
diabetes_X_test  = diabetes.data[-20:]
diabetes_y_train = diabetes.target[:-20]
diabetes_y_test  = diabetes.target[-20:]

regr = linear_model.LinearRegression(copy_X=True, fit_intercept=True)
regr.fit(diabetes_X_train, diabetes_y_train)
print("Coefficients: ", regr.coef_)

print("Mean squared error: ",
                np.mean((regr.predict(diabetes_X_test)-diabetes_y_test)**2))

# Explained variance score: 1 is perfect prediction
# and 0 means that there is no linear relationship
# between X and y.
print("Reg score ", regr.score(diabetes_X_test, diabetes_y_test))
```

Regression Example: Code and Output (2/2)

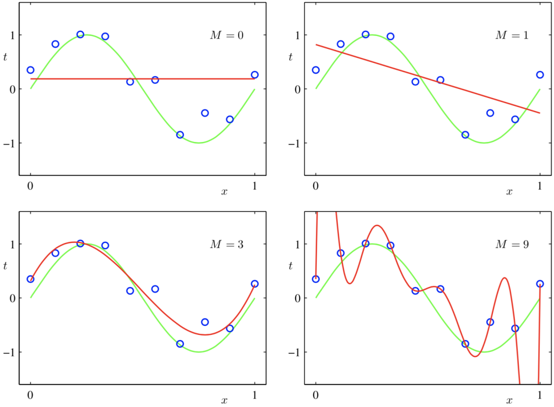
Plotting with the first feature[1]; line = **decision boundary**



Output:
Coeffiecients: [3.03499549e-01 -2.37639315e+02 5.10530605e+02 3.27736980e+02 -8.14131709e+02 4.92814588e+02 1.02848452e+02 1.84606489e+02 7.43519617e+02 7.60951722e+01]
Mean squared error: 2004.5676026898218
Reg score: 0.585

[1] https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html

Overfitting in Regression Task



x-axis = values 0 to 1; y-axis = output of those values in function $sin(2\pi x)$ + random noise
Green line = curve of the function $sin(2\pi x)$ without noise; red = polynomial function that we fit to the training data
source: Bishop, Christopher (2006) "Pattern Recognition and Machine Learning", Springer.

Support Vector Machines (SVM)

Support Vector Machines (SVM)

Definition

SVM is a discriminative classifier formally defined by a separating hyperplane (aka **decision boundary**). In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. The optimal separating hyperplane maximizes the distance to the nearest training points (called the margin). SVMs allow you to use multiple different kernels to find non-linear decision boundaries.

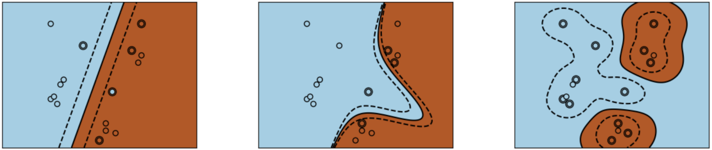
Why maximize margin to training examples?
Because it increases the robustness of the classification algorithm.
This means it is more resistant to noise in the data.

Kernel

Definition

A kernel is a mathematical function of distance that is used to determine the weight of each training example. Kernel methods are algorithms for pattern analysis (=find relations of data points in the dataset) most commonly used in SVMs.

Linear Polynomial Radial Basis Function (RBF)



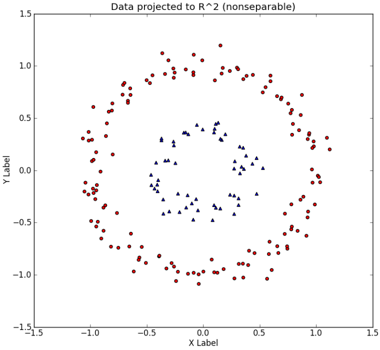
Source: `http://scikit-learn.org/stable/auto_examples/svm/plot_svm_kernels.html`

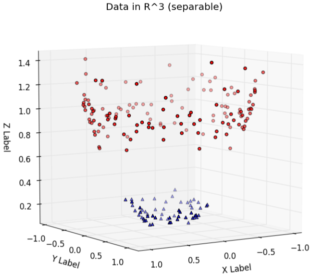
Kernel

Since SVM basically linearly separates data how can it create decision boundaries such as the two non-linear above?

Answer:
Using a mathematical transformation, SVMs move the original data set into a new (usually high dimensional) mathematical space in which the decision boundary is easy to describe/linear.

Transformation





The Kernel Trick

We do not have to manually add features but instead select the kernel function for the SVM that does the "kernel trick".

Kernel trick

Transformation of data into a higher-dimensional space where there is a clear division line to separate data into classes. To do this, kernel functions are used, which usually does not compute the actual coordinates in higher-dimensional feature space but rather the inner products between the input vectors/images of all pairs of data in feature space. This is computationally cheaper.

TECHNISCHE UNIVERSITÄT DRESDEN

Types of Kernels

Each type of kernel is based on a specific kernel function $k(x, x')$ that computes the inner product of two projected vectors

- **Linear**: provides a linear separator with the 'normal' dot product; $k(x, x') = x * x'$
- **Polynomial**: can distinguish curved or nonlinear inputs., where d is the degree of the polynomial (d=1 is similar to a linear transformation), which needs to be manually specified (parameter in sklearn learn); $k(x, x') = ((x * x') + c)^d$
- **Radial Basis Function**: induces space of Gaussian distributions; it calculates the squared Euclidean distance with a kernel coefficient gamma to generate radial areas around training points; $k(x, x') = (exp(-\gamma||x - x'||^2)$

Parameters of SVMs

- **C**: penalty parameter C of the error term
  - tradeoff between how smooth the decision boundary is and how well it classifies examples
  - default = 1.0
  - large C = smoother boundary or more points classified correctly?
- **Gamma**: kernel coefficient for rbf, poly, and sigmoid; defines the influence of a single training example

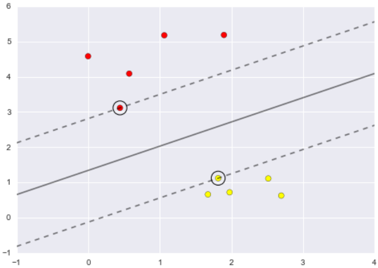
```
import numpy as np
X = np.array([[-1, -1], [-2, -1], [1, 1], [2, 1]])
y = np.array([1, 1, 2, 2])
from sklearn.svm import SVC
clf = SVC(C=1.0,gamma='auto',kernel='rbf')
clf.fit(X, y)
print(clf.predict([[-0.8, -1]])) #Output: [1]
```

Source: http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

What are the support vectors?

Support vectors

Support vectors are the **input vectors** closest to the division line of the data on the basis of which the margin is maximized.



(Dis)Advantages of SMV

Advantages:

- effective in complicated domains with a clear margin of separation between classes
- memory efficient: uses a subset of training points in the decision function (called support vectors)
- flexible: different kernel functions can be specified

Disadvantages:

- do not work well with lots of noise
- slow and prone to overfitting on very large datasets with many features
- do not directly provide probability estimates (expensive to calculate, e.g. using 5-fold cross validation)

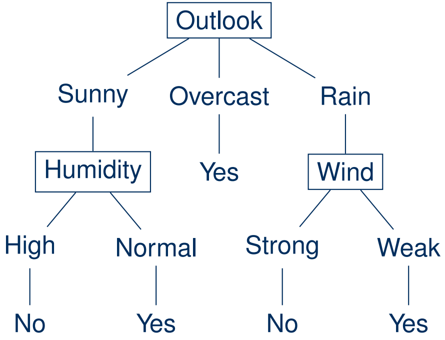
Decision Trees

Decision Trees

Definition

Decision Tree learning is one of the most widely used methods for inductive inference; it is a method to approximate discrete-valued target functions that is robust to noise and capable of learning disjunctive expressions (represent a disjunction of conjunctions on the constraints of the attribute values of instances).

Decision Tree example



Decision Tree for “Playing Tennis on Saturday”

$$(Outlook = Sunny \wedge Humidity = Normal)$$
$$\vee\ (Outlook = Overcast)$$
$$\vee\ (Outlook = Rain \wedge Wind = Weak)$$

$\boxed{\text{Attribute}}$ Value

Source: Mitchell, T. M. (1997). Machine Learning, McGraw-Hill Higher Education. New York.

Parameters of Decision Trees

- **min_sample_split**: the minimum number of samples required to split an internal node (default = 2)
- **criterion**: The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain.

```
class sklearn.tree.DecisionTreeClassifier(criterion=‘gini’, splitter=‘best’,
 max_depth=None, min_samples_split=2, min_samples_leaf=1,
  min_weight_fraction_leaf=0.0, max_features=None, random_state=None,
 max_leaf_nodes=None,  min_impurity_decrease=0.0, min_impurity_split=None,
 class_weight=None, presort=False)
```

Source: http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

Entropy

Definition

Entropy “characterizes the (im)purity of an arbitrary collection of examples.” It measures the homogeneity of examples.

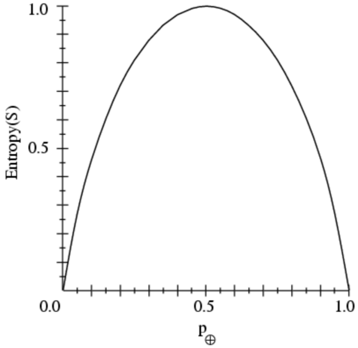
Source: Mitchell, T. M. (1997). Machine Learning, McGraw-Hill Higher Education. New York.

- **S**: collection of training examples
- $p_{\oplus}$: proportion of positive examples in S
- $p_{\ominus}$: proportion of negative examples in S

$Entropy(S) = H(S) = -p_{\oplus} log_2 p_{\oplus} - p_{\ominus} log_2 p_{\ominus}$

14 examples, 9+, 5- => H(S)?

Entropy Overview by Proportion of Positive Examples



Source: Mitchell, T. M. (1997). Machine Learning, McGraw-Hill Higher Education. New York.

Entropy Beyond Binary Classification

The above formula only applies to cases where the task at hand is a binary classification with two potential results/classes for the classifier. In all other cases the following formula applies:

$$H(S) = \sum_{i=1}^{c} -p_i log_2 p_i$$

Where c is the number of target classes and p_i is the proportion of S belonging to class i.

Information Gain

Definition

Information gain "measures how well a given attribute separates the training examples according to their target classification. ... It is the expected reduction in entropy caused by partitioning examples according to this attribute."

Source: Mitchell, T. M. (1997). Machine Learning, McGraw-Hill Higher Education. New York.

$$Gain(S, A) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} H(S_v)$$

$Values(A)$ is the set of all possible values for attribute A
S_v is the subset of S for which Attribute A has value v

Bias-Variance Tradeoff

Necessity to simultaneously minimize two sources of errors that prevents the supervised algorithm from generalizing beyond its training sets.

- **Bias:** High bias (prior assumption) can cause an algorithm to miss the relevant relations between features and target outputs (underfitting)
- **Variance**: error from sensitivity to small fluctuations in the training set. High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs. The smaller the test set, the greater the expected variance. (overfitting)

One way to minimize both: Random Forests = a collection of decision trees whose results are aggregated into one final result (ensemble methods)

(Dis)Advantages of Decision Trees

Advantages:

- simple to understand and interpret (easy to visualize)
- white box model: decision can be explained in boolean logic
- good at handling multi-output problems

Disadvantages:

- prone to overfitting (especially with a large number of features)
- unstable to small variations in the data
- easy to produce biased trees if some class dominates (necessary to balance the dataset prior to fitting)

Review of Lecture 5

- What is linear regression? How does it differ from classification?
- What is an error function?
- How do Support Vector Machines work?
- What is a kernel and what is the kernel trick?
- What are Decision Trees? What are their main advantages?
- What is entropy and how does it relate to information gain?
- What is the bias-variance tradeoff?