Patter Recognition - 6th lab
Linear Classifier and SVM

Viktor Kocur
viktor.kocur@fmph.uniba.sk

DAI FMFI UK

30.3.2020

Linear classifier

Basics

The core principle of a linear classifier is a linear function $f : \mathbb{R}^n \mapsto \mathbb{R}, f(\vec{x}) = \vec{w}^T\vec{x} + b$, where $\vec{x}$ is the feature vector, $\vec{w}$ is the weight vector a b is the bias term.

Classification

If we have two classes ω_1 a ω_2, then we add the feature vector $\vec{x}$ to the class ω_1 if $f(\vec{x}) \geq 0$, and to class ω_2 if $f(\vec{x}) < 0$.

Linear classifier

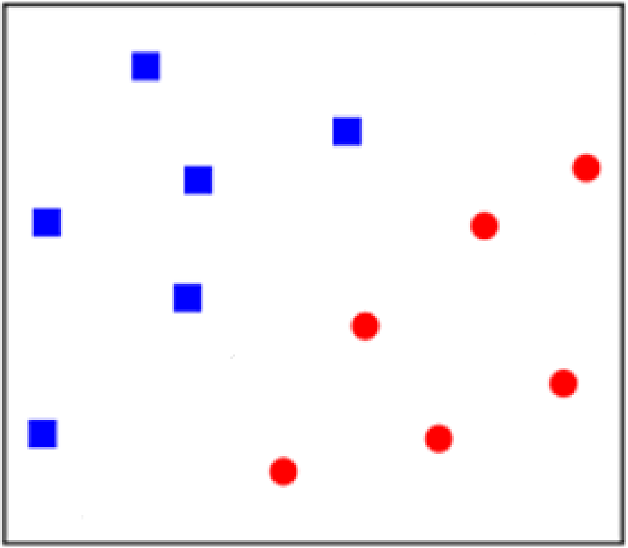
Geometric interpretation

The function f divides the feature space into two areas divided by a hyperplane. Points where $f(\vec{x}) = 0$ lie exactly on this hyperplane.

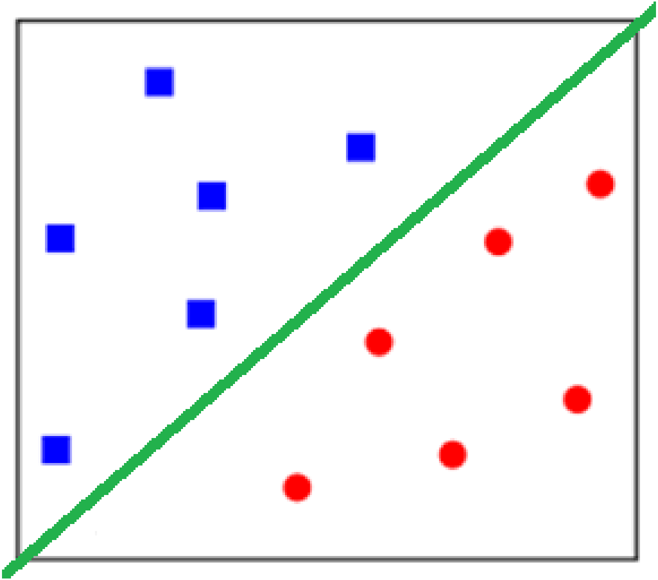
Training

We want to find the parameters of our classifier so that the hyperplane separates the training set the best.

Linear classifier



Linear classifier



Training

Training

To train the classifier we will need the so-called training data. E.g. pairs of feature vectors $\vec{x}$ with labels $y \in \{0, 1\}$ determined by the correct class. Our goal is for the classifier to work well on the training data.

Regularization

Sometimes we want a classifier which is not the best one possible on the training data. Instead we want one that can generalize well. This kind of approach is called regularization.

Cost function - I

Cost function

We obtain a good classifier by creating a cost function $C : \mathbb{R}^{n+1} \mapsto \mathbb{R}$, $C(b, \vec{w})$, which has a global minimum for parameters which separate the classes the best. Training is then an optimization task.

Cost function - II

Simplification

Since the bias term b complicates things a bit we will use new notation $\vec{\theta} = (b, \vec{w})$ and $\vec{X} = (1, \vec{x})$. Such a change enables us to use the expression: $f(\vec{X}) = \vec{\theta}^T \vec{X}$.

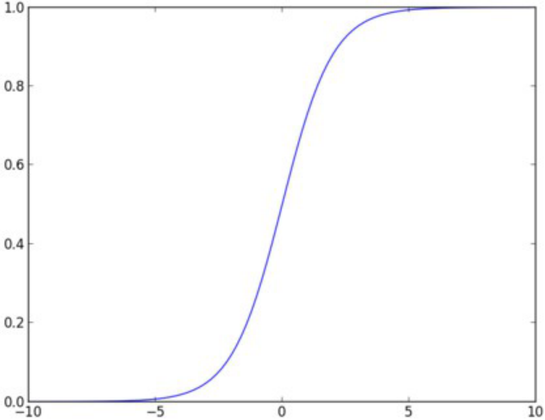
Sigmoid

We will use the sigmoid function: $\sigma(z) = \frac{1}{1+e^{-z}}$.

Sigmoid - derivative

$\sigma(z)' = \sigma(z)(1 - \sigma(z))$.

Sigmoid



Cost function - III

Simplification

Let us consider a function: $h_\theta = \sigma(f(\vec{x}))$.

Cost function - binary crossentropy

$$J(\vec{\theta}) = \frac{1}{m}\sum_{i=1}^{m}\left(-y^{(i)}log(h_\theta(\vec{x}^{(i)})) - (1-y^{(i)})log(1-h_\theta(\vec{x}^{(i)}))\right)$$

Cost function - derivative

$$\frac{\partial J}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(\vec{x}^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

Optimizataion

Gradient descent

$\theta_i := \theta_i - \eta \frac{\partial J}{\partial \theta_i}$

Optimization in reality

Usually the optimization is performed using a more sophisticated algorithm such as SGD, or methods based on the Hess matrix.

Optimization in Matlab

x = fminunc(fun,x0) - finds the optimal parameters where the function fun is minimal. Since this function uses iterative methods it is necessary to add initial value x0.

Optimization

Exercise

Check the LinearClassifier.m script.

Exercise

Finish the function costFunction using the cost function we introduced few slides back.

Linear classifier - Matlab

Regularization

We add a regularization term to the cost function: $C_R(\vec{\theta}) = C(\vec{\theta}) + R(\vec{\theta})$. For example $R(\vec{\theta}) = \sum_{i=2}^{n} \theta_i^2$, or $\sum_{i=2}^{n} |\theta_i|$

fitclinear

Mdl = fitclinear(x,y) - returns a classification model Mdl for feature vectors which are in rows of matrix x and correct classes in y. This function can perform SVM and logistical regression. Check out help. Regularization is used by default.

Mdl.predict

Mdl.predict(x) - returns the class for given feature vector

Linear classifier - Matlab

Mdl.Beta, Mdl.Bias

Mdl.Beta - returns our weight vector $\vec{w}$. Mdl.Bias - returns the bias term b.

Exercise

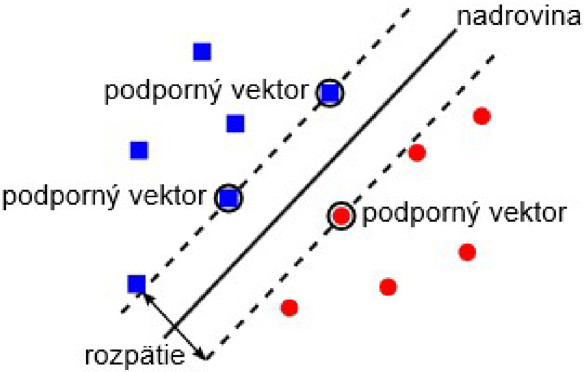
Into the image (gscatter) with data from ex2data1.txt add the line which separates the data for fitcliniear classifier. You can use plot or refline commands. Note for refline we use:

$$\beta_1 \cdot x_1 + \beta_2 \cdot x_2 + bias = 0 \iff x_2 = m \cdot x_1 + b$$

$$x_2 = -\frac{\beta_1}{\beta_2} \cdot x_1 - \frac{bias}{\beta_2}$$

$$m = -\frac{\beta_1}{\beta_2}, b = -\frac{bias}{\beta_2}$$

Idea of SVM


nadrovina
podporný vektor
podporný vektor
podporný vektor
rozpätie

SVM

Basics

SVM finds support vectors and attempts to find paramaters so that the gap between the classes is the widest. This is achieved by trying to find parametrization so that $\vec{w}^T \vec{x} + b = \pm 1$ for the support vectors.

Kernels

Data is usually not linearly separable. Therefore it is necessary to transform the feature space using so-called kernels. Kernels are functions $\phi : \mathbb{R}^n \mapsto \mathbb{R}^m$, for which a function k exists so that: $k(x_i, x_j) = \phi(x_i)\phi(x_j)$. SVM then finds a linear classifier in the new space $\mathbb{R}^m$.

SVM

fitcsvm

SVMMdl = fitcsvm(X,y) - returns an SVM model on features X and classes y.

fitcsvm

SVMMdl = fitcsvm(X,y, 'KernelFunction',nazov, 'KernelScale', 'auto') - returns an SVM with kernel trick. Note: do not forget the scale.

SVM - Úloha

showSVM

showSVM(SVMMdl, X, y) - displays the SVM model for 2D data X,y (this is an m-file in the zip)

Exercise

Display an SVM with various kernels. Check out what happens when you do not set the KernelScale.