Data Mining: Lecture 6

Classification: Support Vector machines and Kernel Trick

S. Nõmm

[1]Department of Software Science, Tallinn University of Technology

10.10.2023

Separability

Linear separability

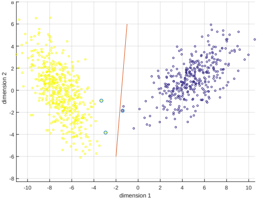
Two classes are said to be linearly separable in $\mathbb{R}^n$ -if there exists a hyperplane dividing the space into two subspaces such that all the elements of the first class belong to one subspace and the elements of the second class belong to the other subspace.

Or

-if there exist n - dimensional vector a and scalar b such that for the elements of one class $x^T a > b$ and for the elements of the second class $x^T a < b$

Separability

- If two classes are linearly separable it is possible to construct two hyperplanes, parallel to the "separating" hyperplane, such that first hyperplane would contain at least one point of the first class and second hyperplane will contain at least one point of the second class.
- The training data points belonging to these hyperplanes are referred as support vectors and the distance between the hyperplanes is referred as margin.
- In order to determine maximum margin hyperplane nonlinear programming optimization is required. First margin is expressed as the function of the coefficients of separating hyperplane. Second optimization problem is solved.

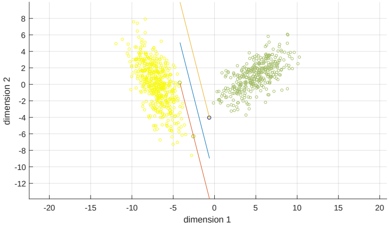


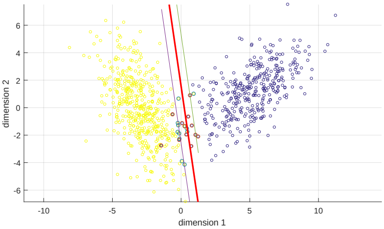
Maximum margin hyperplane

- For the hyperplane $x^T a + b$ vector $a = (a_1, \ldots, a_n)$ is n - dimensional vector representing the normal direction to the hyperplane.
- Then the distance (margin) from the separating hyperplane to the hyperplanes containing points of each class (see previous slide) would be $M = ||a||^{-1}$.
- The optimization problem may be stated in terms of finding vector a that would maximize margin

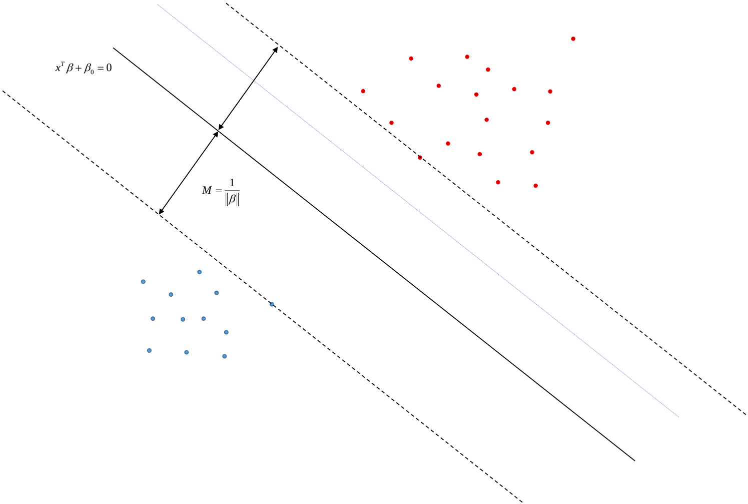
Hard and Soft Margin cases

Hard margin case is depicted by the left figure and soft margin by the figure on the right side.





Linear Case: Hard-Margin case



Linear Case: Hard-Margin case

- Let N pairs (x_i, y_i) where $i = 1, \dots N$ constitute the training data; $x_i \in \mathbb{R}$ and $y_i \in \{-1, 1\}$.
- Define the hyperplane as $\{x : f(x) = x^T b_1 + b_0 = 0\}$, where $||b|| = 1$.
- Classification rule induced by $f(x)$ is

$$G(x) = \text{sign}[x^T b_1 + b_0].$$

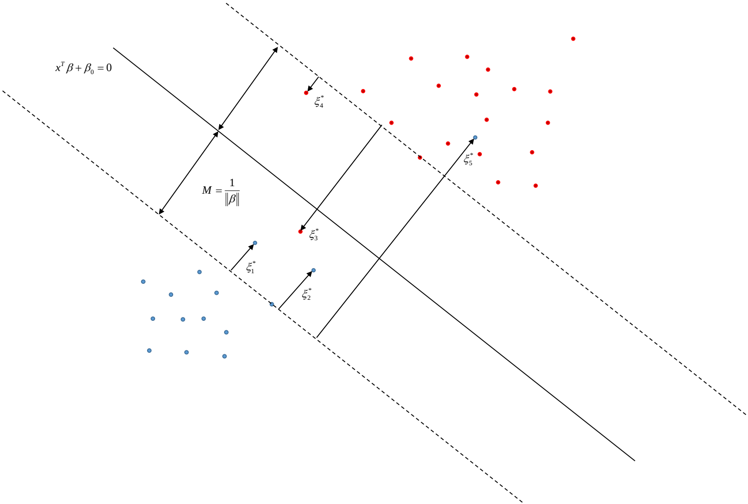
- Classes are separable $\Rightarrow \exists f(x) = x^T b_1 + b_0 : y_i f(x_i) > 0, \forall i$.
- One is able to find the hyperplane that creates the biggest margin between the training points leads following optimization problem:

$$\max_{b_1, b_0, ||b||=1} M$$
$$\text{subject} \quad \text{to} \quad y_i(x^T b_1 + b_0) \geq M, i = 1, \dots, N,$$

or in a more convenient form:

$$\min_{b_1, b_0} ||b||$$
$$\text{subject} \quad \text{to} \quad y_i(x^T b_1 + b_0) \geq 1, i = 1, \dots, N; \quad M = 1/||b||$$

Linear Case: Soft-Margin case



Linear Case: Soft-Margin case

If the classes in training data overlap then we are talking about soft margin case.

- One of the possible way is to maximize M, whereas it is allowed to some points to be on the "wrong" side of the plane.
- Define the slack variables $\xi = (\xi_1, \dots, \xi_N)$
- The first way to modify optimization problem is $y_i(x^T b_1 + b_0) \geq M - \xi_i$. (results in a non-convex optimization problem).
- The second way is $y_i(x^T b_1 + b_0) \geq M(1 - \xi_i)$. (results in a convex optimization problem).
- $\forall i, \xi_i \geq 0, \sum_{i=1}^{N} \xi_i$ is limited by some constant
- The second way leads to the standard vector classifier.

Soft-Margin case

- For the case when classes are allowed to overlap some points are allowed to be on the 'wrong' side of the hyperplane. The distances of these points from their margin are denoted as ξ_i.
- The first way to describe the constraint is

$$y_i(x_i^T b_1 + b_0) \geq M - \xi_i$$

  In this form constraint described overlap in actual distance but lead non convex optimization problem.
- The second way is

$$y_i(x_i^T b_1 + b_0) \geq M(1 - \xi_i)$$

  In this form overlap is described in relative distance but lead convex optimization problem. ξ is the proportional amount by which the prediction $f(x_i) = x_i^T b_1 + b_0$ is on the wrong side of its margin.
- Bounding sum of ξ_i allows to bound the total proportion amount by which the predictions fall into 'wrong' side of their margin. Misclassifications occur when $\xi_i > 1$.

Lagrangian theory

- The goal is to find separating hyperplane maximizing the margin.
- This problem may be seen as the minimization of a function with additional constraints described by linear equations.
- Denote function to be minimized as $f(\theta)$ and let constraints be presented by $h_i(\theta) = 0$, $i = 1, \dots, m$. Lagrangian function is defined as follows

  $$L(\theta, \beta) = f(\theta) + \sum_{i=1}^{m} \beta_i h_i(\theta)$$

  coefficients β_i are called Lagrange multipliers.
- Necessary condition for point θ^* to be a minimum of $f(\theta)$ subject to $h_i(\theta) = 0$, $i = 1, \dots, m$ is

  $$\begin{aligned} \frac{\partial L(\theta^*, \beta^*)}{\partial \theta} &= 0 \\ \frac{\partial L(\theta^*, \beta^*)}{\partial \beta} &= 0 \end{aligned}$$

  if L is convex function then this condition is also sufficient.

Example

- Maximize: $f(x_1, x_2) = 1 - x_1^2 - x_2^2$
  Subject to $h(x_1, x_2) = x_1 + x_2 - 1 = 0$
- Computed in class.
- solution $\mathbf{x}^* = (0.5, 0.5)$ and $\beta = 1$.

Objective functions and corresponding Lagrangians

- Hard margin case

$$\min_{\theta,b} \frac{1}{2}||\theta||^2$$
$$y_i(\theta^T X_i + b) \geq 1, \forall i$$

- Lagrangian for the hard margin SVM is:

$$L(\theta, b, \alpha) = \frac{1}{2}||\theta||^2 - \sum_{i=1}^{n} \alpha_i(y_i(\theta^T X_i + b) - 1).$$

- Soft margin case

$$\min_{\theta,b} \frac{1}{2}||\theta||^2 + C\sum_i \xi_i$$
$$y_i(\theta^T X_i + b) \geq 1 - \xi_i, \forall i$$
$$\xi_i \geq 0 \forall i$$

- Lagrangian for the soft margin SVM is:

$$L(\theta, b, \xi, \alpha, \beta) = \frac{1}{2}||\theta||^2 + C\sum_{i=1}^{n} \xi i - \sum_{i=1}^{n} \mu_i \xi_i - \sum_{i=1}^{n} \alpha_i[y_i(\theta^T X_i + b) - 1 + \xi_i]$$

Solution

- Lagrangian (Wolfe) dual objective function.

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle X_i \cdot X_j \rangle$$

subject to $C \geq \alpha \geq 0$, $i = 1, \ldots, n$.

Support Vector Machines: nonlinear case

- Let the decision boundary between two classes is given by

$$8(x_1 - 1)^2 + 50(x_2 - 2)^2 = 1$$

  may be easily rewritten as

$$8x_1^2 - 16x_1 + 50x_2^2 - 200x_2 + 207 = 0$$

- It may be expressed linearly in terms of four variables $z_1 = x_1^2$, $z_2 = x_1$, $z_3 = x_2^2$ and $z_4 = x_2$ as

$$8z_1 - 16z_2 + 50z_3 - 200z_4 + 207 = 0$$

- The only drawback of this approach is that it requires to perform transformation explicitly.

The Kernel trick

- Lagrangian (Wolfe) dual objective function.

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle X_i \cdot X_j \rangle$$

  subject to $C \geq \alpha \geq 0$, $i = 1, \ldots, n$.

- Observe that this problem may be solved not in terms of data points but in terms of their dot products. This leads the idea to define similarity function on the transformed representation with use of so called kernel function.

$$K(X_i, X_j) = \Phi(X_i) \cdot \Phi(X_j)$$

  In this case Lagrangian (Wolfe) dual objective function will take shape

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(X_i, X_j).$$

- Corresponding equation of the decision boundary $\Phi(X)^T a + b$

Kernel functions

Kernel function is real valued function of two arguments $k(x, x') \in \mathbb{R}$ for $x, x' \in \mathcal{X}$. Typically the function is

- symmetric $k(x, x') = k(x', x)$
- nonnegative $k(x, x') \geq 0$.

It may be interpreted as distance function.

Examples I:RBF

- RBF (radial base function) kernels:
  - Squared exponential kernel or Gaussian kernel

$$k(x,x') = \exp\Big(-\frac{1}{2}(x-x')\Sigma^{-1}(x-x')\Big)$$

  - if Σ is spherical then gaussian kernel becomes RBF:

$$k(x,x') = \exp\Big(-\frac{||x-x'||^2}{2\sigma^2}\Big)$$

Examples II: comparing documents

- Cosine distance:

$$k(x_i, x_{i'}) = \frac{x_i^T x_{i'}}{||x_i||_2 ||x_{i'}||_2}$$

- apply TF-IDF representation

$$\begin{aligned} \mathrm{tf}(x_{ij}) &\triangleq \log(1 + x_{i,j}) \\ \mathrm{idf}(j) &\triangleq \log \frac{N}{1 - \sum_{i=1}^{N} \mathbb{I}(x_{ij} > 0)} \end{aligned}$$

- TF-IDF representation is given by:

$$\mathrm{tf - idf(x_i)} \triangleq \mathrm{[tf(x_i)^T \times idf(j)]_{j=1}^V}$$

- corresponding kernel function is

$$k(x_i, x_{i'}) = \frac{\phi(x_i)^T \phi(x_{i'})}{||\phi(x_i)||_2 ||\phi(x_{i'})||_2}$$

where $\phi(x) = \mathrm{tf - idf(x)}$.

Examples III: Mercer (positive defined kernels)

- Some methods require that the kernel function satisfy the condition that Gram matrix:

$$K = \begin{pmatrix} k(X_1, X_1) & \dots & k(X_1, X_N) \\ & \vdots & \\ k(X_N, X_1) & \dots & k(X_N, X_N) \end{pmatrix}$$

  be positive define for any set of inputs. kernels satisfying this condition are called Mercer kernels or positive definite kernels.

- Mercer's theorem states, that if the gram matrix is positive define eigenvector decomposition may be computed as follows: $K = U^T \Lambda U$.
- An element of K is therefore $k_{i,j} = (\Lambda^{\frac{1}{2}} U_{:i})^T (\Lambda^{\frac{1}{2}} U_{:j})$. Let $\phi(x_i) = \Lambda^{\frac{1}{2}} U_{:i}$, which leads $k_{ij} = \phi(x)^T \phi(x')$.
- The entries of the kernel matrix can be computed by performing an inner product of some feature vectors. In general if the kernel is Mercer then there exists a function ϕ mapping $x \in \mathcal{X}$ to $\mathbb{R}^D$ such that: $k(x, x')$. Where ϕ depends on eigenfunctions of K.

Examples III: Mercer (positive defined kernels)

If K_1 and K_2 are Mercer's kernels then

- $K_1(x_1, x_2) + K_2(x_1, x_2)$,
- $aK_1(x_1, x_2)$, $a \in \mathbb{R}$,
- $K_1(x_1, x_2)K_2(x_1, x_2)$
- $\exp(K_1(x_1, x_2))$,

are Mercer's kernels

The kernel trick: most popular kernel functions

K should be a symmetric positive define function.

- n -th Degree polynomial $K(x_i, x_j) = (1 + x_i \cdot x_j)^n$.
- Radial basis $K(x_i, x_j) = \exp(-\gamma ||x_i - x_j||^2)$.
- Sigmoid kernel $K(x_i, x_j) = \tanh(k_1 x_i \cdot x_j + k_2)$.

The Kernel trick

- Instead of defining feature vector in terms of kernels one can work with original feature vectors x.
- This approach requires one to modify algorithm so that it replaces all inner products of the form $\langle x, x' \rangle$ with the call to the kernel function $k(x, x')$. This approach is referred as <u>kernel trick</u>.
- one can kernelize many ML algorithms:
- Kernelized nearest neighbor classifier: $||x_i - x_{i'}||_2^2 = \langle x_i, x_j \rangle + \langle x'_i, x'_j \rangle - 2\langle x_i, x'_j \rangle$. In such form nearest neighbour classifier may be applied to non structured data objects.
- Kernelized: k-medoids algorithm, ridge regression,PCA etc.

Cross validation

- Non-exhaustive do not use all possible ways of splitting into training and validation sets
  - k - fold.
  - Holdout.
  - Repeated random sub-sampling.
- Exhaustive: use all possible ways to divide the data set into training and validation sets
  - Leave p-out cross validation.
  - Leave one out cross validation.

Cross validation: k- fold validation

- Divide the training data (after removing test data) randomly into k - folds.
- Perform following k experiments:
  - Compose the training data by concatenating k-1 folds leaving one fold out.
  - Train the model on those k-1 folds
  - Test it on the left-out fold
  - Record the result
- Report the average of the k experiments.

Let us remind the main idea of Cross Validation

- The method to estimate the expected extra-sample error $\mathcal{E} = E[L(Y, \hat{f}(X))]$ (average generalized error) when the method $\hat{f}(X)$ is applied to and independent test sample from the joint distribution of X an Y (L denots loss function here.)
- Cross-validation estimate of prediction error is given by:

$$\mathcal{E}_{CV} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}^{-k(i)}(x_i)).$$

- Usually 5 or 10 fold cross validation is recommended.

Cross Validation within Machine Learning Work-flow

- Up to a present time we have used synthetic sets of a very small power, treating them as the samples.
- For the real life applications when one have the sample only and not entire population this may lead to serious errors.
- One possible way to fix the problem is to perform feature selection within the cross validation loop. (Point to discuss!!!)

Hastie & Tibshirian view on cross validation

- Consider to study in detail section $7.10.2$ The Elements of statistical learning.
- Classification problem with a large number of predictors.
- What would be the strategy to implement ML work flow?

Example p. 245

- $N = 50$ samples, binary case, two equal sized classes.
- Let the power of feature set be $p = 5000$, each feature normally distributed and independent of class labels.
- True error rate for any classifier is 0.5
- Let us suppose that 100 predictors is chosen.
- 1-nearest neighbour classifier was chosen.
- 50 simulations will result in cross validation error of 0.03, whereas true error rate is 0.5
- Leaving samples out after the feature selection does not mimic correctly the application of the classifier to a previously unseen data.

H & T suggest that this is the (correct) way :$\tilde{)}$

- Divide the data set into K cross-validation folds.
- For each fold k perform:
- Use all the folds except the fold k to perform the feature selection and model training.
- Use fold k for model validation.
- Use the results for each k to compute error estimates.

What is the drawback of cross validation?