Machine learning algorithms for inverse problems
Kernel Methods and Support Vector Machines (SVM)
Lecture 9

Literature

Used literature:

- Christopher M. Bishop, *Pattern recognition and machine learning*, Springer, 2009.
- Ian Goodfellow, Yoshua Bengio and Aaron Courville, *Deep Learning*, MIT Press, 2016, http://www.deeplearningbook.org
- Miroslav Kurbat, *An Introduction to Machine Learning*, Springer, 2017.

Dual representations and kernel methods

- A lot of parameter identification problems can be considered using dual representation.
- Prediction in this dual represenation is based on linear combination of a kernel function evaluated at the training data points.
- Usually, the kernel function is given by the equation

$$k(x,x') = \varphi(x)^T \varphi(x')$$

  where $\varphi(x)$ is a test function for the model.
- The kernel function is symmetric s.t.

$$k(x,x') = k(x',x)$$

- The easest test function is $\varphi(x) = x$, then the linear kernel is

$$k(x,x') = \varphi(x)^T \varphi(x') = x^T x'$$

- The concept of kernel functions was introduced by Aizerman et al. in 1964, then forgotten and was re-introduced again by Boser et.al in 1992 to use in support vector machines (SVM)

Dual representations and kernel methods

Linear models for regression and classification can be reformulated in terms of dual representation. Recall the linear regression model where we have minimized the following regularization functional:

$$
\begin{aligned}
J(\omega) &= \frac{1}{2}\sum_{n=1}^{N}(\omega^T\varphi(x_n) - t_n)^2 + \frac{\lambda}{2}\omega^T\omega \\
&= \frac{1}{2}\|\omega^T\varphi(x) - t\|_2^2 + \frac{\lambda}{2}\|\omega\|_2^2 = \frac{1}{2}(\omega^T\varphi(x) - t)^T(\omega^T\varphi(x) - t) + \frac{\lambda}{2}\omega^T\omega.
\end{aligned}
\tag{1}
$$

where $\lambda \geq 0$ is the regularization parameter.

Dual representations and kernel methods

Taking $J'_\omega(\omega) = 0$ we get optimal ω (here, n-th row of the design matrix Φ is $\varphi(x_n)^T$):

$$
\begin{aligned}
0 = J'_\omega(\omega) &= (\omega^T\varphi(x) - t)\varphi(x) + \lambda\omega, \\
\omega = -\frac{1}{\lambda}(\omega^T\varphi(x) - t)\varphi(x) &= \underbrace{-\frac{1}{\lambda}\sum_{n=1}^{N}(\omega^T\varphi(x_n) - t_n)}_{a_n}\varphi(x_n) \qquad (2)\\
&= \sum_{n=1}^{N} a_n\varphi(x_n) := \Phi^T a.
\end{aligned}
$$

Here, n-th row of the design matrix Φ is $\varphi(x_n)^T$.

Dual representations and kernel methods

From (2) follows that $\omega = \Phi^T a$. Then $\omega^T = a^T \Phi$. Substitute these expressions into (3):

$$\begin{aligned} J(\omega) &= \frac{1}{2}\|\omega^T\varphi(x) - t\|_2^2 + \frac{\lambda}{2}\|\omega\|_2^2 = \frac{1}{2}(\omega^T\varphi(x) - t)^T(\omega^T\varphi(x) - t) + \frac{\lambda}{2}\omega^T\omega \\ &= \frac{1}{2}[(\omega^T\varphi(x))^T\omega^T\varphi(x) - 2\omega^T\varphi(x)t + t^Tt] + \frac{\lambda}{2}\omega^T\omega \\ &= \frac{1}{2}[\Phi(\Phi^T a)(a^T\Phi)\Phi^T - 2(a^T\Phi)\Phi^T t + t^Tt] + \frac{\lambda}{2}(a^T\Phi)(\Phi^T a). \end{aligned} \tag{3}$$

Now we define the $N \times N$ symmetric kernel matrix $K = \Phi\Phi^T$, $K_{nm} = \varphi(x_n)^T\varphi(x_m) = k(x_n, x_m)$ and rewrite the above equation in the terms of kernel:

$$\begin{aligned} J(a) &= \frac{1}{2}a^TKKa - a^TKt + \frac{1}{2}t^Tt + \frac{\lambda}{2}a^TKa \\ &= \frac{1}{2}\|a^TK - t\|_2^2 + \frac{\lambda}{2}K\|a\|_2^2 \end{aligned} \tag{4}$$

Dual representations and kernel methods

Now we find optimal a by taking $J'_a(a) = 0$:

$$\begin{aligned} J(a) &= \frac{1}{2}\|a^T K - t\|_2^2 + \frac{\lambda}{2} K\|a\|_2^2, \\ 0 = J'_a(a) &= (a^T K - t)K + \lambda K a, \\ a &= (K + \lambda I_N)^{-1} t. \end{aligned} \tag{5}$$

As soon as we have found optimal $a = (K + \lambda I_N)^{-1} t$, it can be substituted in the linear regression model $y(x) = \omega^T \varphi(x)$, where first we will substitude $\omega^T = a^T \Phi$ and then use definition of a. We have following equation to predict the new point x:

$$y(x) = \omega^T \varphi(x) = (a^T \Phi)\varphi(x) = k(x)^T (K + \lambda I_N)^{-1} t, \tag{6}$$

where $k_n(x) = k(x_n, x)$. We observe that the solution is obtained in terms of kernel function.

Constructing kernels

Let us discuss how to construct the valid kernel functions.

- 1. The first approach is to construct kernel from it's definition

$$k(x,x') = \varphi(x)^T\varphi(x') = \sum_{i=1}^{M}\varphi_i(x)\varphi_i(x')$$

- 2. Another approach is to construct the kernel functions directly. In this case we should be sure that the function which we decided to take is a valid kernel.

Example

take a s kernel function

$$k(x,z) = (x^Tz)^2.$$

Then taking $x = (x_1, x_2), z = (z_1, z_2)$ we can write:

$$k(x,z) = (x^Tz)^2 = (x_1z_1 + x_2z_2)^2 = x_1^2z_1^2 + 2x_1z_1x_2z_2 + x_2^2z_2^2$$
$$= (x_1^2, \sqrt{2}x_1x_2, x_2^2)(z_1^2, \sqrt{2}z_1z_2, z_2^2)^T = \varphi(x)^T\varphi(z).$$

Constructing kernels

- A necessary and sufficient condition for a function $k(x, x')$ to be a valid kernel is that the Gram matrix $K = \Phi\Phi^T, K_{nm} = \varphi(x_n)^T \varphi(x_m) = k(x_n, x_m)$ should be positive semidefinite for all choices of the set $\{x_n\}$. Semidefinite K: $x^T K x \geq 0 \forall x \in R^n$.
- One of the possible techniques of constructing kernels is to use simpler valid kernels $k_1(x, x'), k_2(x, x')$ as building blocks to construct a new one kernels:

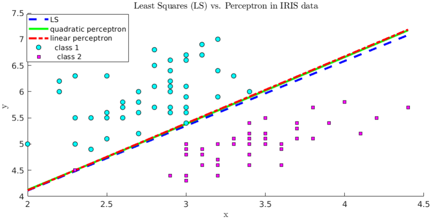
$$\begin{aligned}
k(x, x') &= Const \cdot k_1(x, x'), Const > 0, \\
k(x, x') &= P(k_1(x, x')), P - \text{polynomial} \\
k(x, x') &= f(x) k_1(x, x') f(x'), \\
k(x, x') &= e^{k_1(x, x')}, \\
k(x, x') &= k_1(x, x') + k_2(x, x'), \\
k(x, x') &= k_1(x, x') \cdot k_2(x, x'), \\
k(x, x') &= x^T A x', A - \text{ s.p.semid.}
\end{aligned} \tag{7}$$

Constructing kernels

Popular valid kernels are (we can use rules (7) to prove that they are valid):

$$
\begin{aligned}
k(x,x') &= (x^T x' + const.)^M, \quad const. > 0 \\
k(x,x') &= e^{-\|x-x'\|^2/2\sigma^2} - \text{ (Gaussian kernel)}, \\
k(x,x') &= p(x)p(x'), p(x) - \text{mapping}, \\
k(x,x') &= \sum_i p(x|i)p(x'|i)p(i), \\
& \qquad p(i) = const. > 0, p(x|i), p(x'|i) - \text{probability distrib.}, \\
k(x,x') &= \int p(x|z)p(x'|z)p(z)dz, z - \text{contin. latent variable},
\end{aligned}
$$

SVM: maximum margin classifiers



Let us consider again classification of 2 classes using the linear model

$$y(x) = \omega^T \varphi(x) + b,$$

where $\varphi(x)$ is the test functions and b is the bias. The training datasets consist of $x_1, ..., x_N$ with corresponding target values $t \in \{-1, 1\}$. New datapoints are classified by the sign of $y(x)$.

We already know that it can be a lot of solutions to this problem dependening of the initial guess for weights ω and bias b, see Figure. The best one solution is such that which gives the smallest generalization error.

SVM: maximum margin classifiers

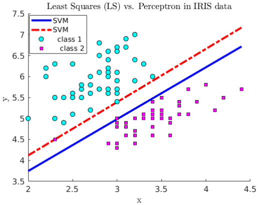


Figure: The margin is the smallest (perpendicular) distance between red and blue lines. Support vectors are points in classes (one blue and one magenta point) crossing the red and blue lines.

SVM approach finds the margin or the smallest distance between the decision boundary and any of the samples. We want to find maximal margin or maximal distance between the decision boundary and the closest of the data sets.

SVM: maximum margin classifiers

Let for point closest to the surface we set

$$t_n(\omega^T \varphi(x_n) + b) = 1 \tag{8}$$

Then all data points willl satisfy

$$t_n(\omega^T \varphi(x_n) + b) \geq 1 \tag{9}$$

By definition, there will be at least two active constraints satisfying (9). The optimization problem is to find optimal weights such that

$$\min_{\omega,b} \frac{1}{2} \|\omega\|_2^2 \tag{10}$$

subject to constrains given by (9).

SVM: maximum margin classifiers

For solution of constrained optimization problem we will construct the following Lagrangian:

$$L(\omega, b, \lambda) = \frac{1}{2}||\omega||_2^2 - \sum_{n=1}^{N} \lambda_n (t_n(\omega^T \varphi(x_n) + b) - 1) \tag{11}$$

where $\lambda_n \geq 0$ is the Lagrangian multiplyer.
We have the minus sign in the Lagrangian since we minimize with respect to ω and b and maximize with respect to $\lambda = (\lambda_1, ..., \lambda_N)^T$. We compute optimality conditions:

$$\begin{aligned}
0 &= L'_\lambda(\omega, b, \lambda) = -(\sum_{n=1}^{N} t_n(\omega^T \varphi(x_n) + b) - 1), \\
0 &= L'_\omega(\omega, b, \lambda) = \omega - \sum_{n=1}^{N} \lambda_n t_n \varphi(x_n) \Longrightarrow \omega = \sum_{n=1}^{N} \lambda_n t_n \varphi(x_n), \qquad (12) \\
0 &= L'_b(\omega, b, \lambda) = -\sum_{n=1}^{N} \lambda_n t_n.
\end{aligned}$$

SVM: maximum margin classifiers

Now we construct a new Lagrangian by substituting (12) into (11) and using the fact that $\sum_{n=1}^{N} \lambda_n t_n = 0$ to get:

$$\begin{aligned}\tilde{L}(\lambda) &= -\sum_{n=1}^{N} \lambda_n (t_n ((\sum_{m=1}^{N} \lambda_m t_m \varphi(x_m))^T \varphi(x_n) + b) - 1) \\ &= -\sum_{n=1}^{N}\sum_{m=1}^{N} \lambda_n \lambda_m t_n t_m \varphi(x_m)^T \varphi(x_n) - \sum_{n=1}^{N} \underbrace{\lambda_n t_n}_{=0} b + \sum_{n=1}^{N} \lambda_n\end{aligned} \tag{13}$$

We get finally:

$$\begin{aligned}\tilde{L}(\lambda) &= \sum_{n=1}^{N} \lambda_n - \sum_{n=1}^{N}\sum_{m=1}^{N} \lambda_n \lambda_m t_n t_m \varphi(x_m)^T \varphi(x_n) \\ &= \sum_{n=1}^{N} \lambda_n - \sum_{n=1}^{N}\sum_{m=1}^{N} \lambda_n \lambda_m t_n t_m k(x_n, x_m).\end{aligned} \tag{14}$$

SVM: maximum margin classifiers

To classify new data points in kernel perceptron algorithm, we now determine the sign of

$$t_n y(x_n) = t_n(\omega^T \varphi(x_n) + b) = t_n(\sum_{n=1}^{N} \lambda_n t_n k(x, x_n) + b). \qquad (15)$$

The equation above is obtained using the optimality condition $L'_\omega(\omega, b, \lambda) = 0$ from what follows that $\omega = \sum_{n=1}^N \lambda_n t_n \varphi(x_n)$. The following 3 conditions should hold:

$$\begin{aligned} \lambda_n &\geq 0, \\ t_n y(x_n) - 1 &\geq 0, \\ \lambda_n(t_n y(x_n) - 1) &= 0. \end{aligned}$$

From the last condition we observe that eather $\lambda_n = 0$ or $t_n y(x_n) - 1 = 0$. Any data point for which $\lambda_n = 0$ will not be presented in the sum (15). Thus, this point will not play roll for prediction. The remaining data points are called support vectors since they satisfy the condition $t_n y(x_n) = 1$. These points will lie on the maximum margin hyperplanes.

SVM: separation of overlapping classes

- We will modify SVM such that we will allow some of class points to be missclassified
- We will add penalty which will increase the distance from the decision boundary
- In order to do this we introduce variables $\xi_n, n = 1, ...N$, N is number of training data points.
- Variables $\xi_n, n = 1, ...N$ are defined as: $\xi_n = 0$ for all points which are on the margin (decision) boundary or inside it. $\xi_n = |t_n - y(x_n)|$ for all other points. Then the point which is on the decision boundary $y(x_n) = 0$ will have $\xi_n = 1$ and points where $\xi_n > 1$ will be missclassified.

  Then instead of (9):

$$t_n y(x_n) \geq 1, n = 1, ..., N$$

  we will have

$$t_n y(x_n) \geq 1 - \xi_n, n = 1, ..., N, \xi_n \geq 0. \qquad (16)$$

SVM: separation of overlapping classes

The goal now is to minimize the following functional subject to constrains given by (16):

$$\min_{\omega,b} \frac{1}{2}||\omega||_2^2 + C \sum_{n=1}^{N} \xi_n \tag{17}$$

with the $C = const. > 0$ which controls the slack variable and the margin and can be considered as a regularization parameter.
For solution of constrained optimization problem we will construct the following Lagrangian:

$$\begin{aligned} L(\omega, b, \xi, \lambda, \mu) &= \frac{1}{2}||\omega||_2^2 + C \sum_{n=1}^{N} \xi_n \\ &- \sum_{n=1}^{N} \lambda_n (t_n(\omega^T \varphi(x_n) + b) - 1 + \xi_n) - \sum_{n=1}^{N} \mu_n \xi_n \end{aligned} \tag{18}$$

where $\lambda_n, \mu_n \geq 0$ are Lagrangian multiplyers.

SVM: separation of overlapping classes

KKT conditions will be for $n = 1, ..., N$:

$$\begin{aligned} \lambda_n &\geq 0, \\ t_n y(x_n) - 1 + \xi_n &\geq 0, \\ \lambda_n(t_n y(x_n) - 1 + \xi_n) &= 0, \\ \mu_n &\geq 0, \\ \xi_n &\geq 0, \\ \mu_n \xi_n &= 0. \end{aligned}$$

SVM: separation of overlapping classes

Optimality conditions will be:

$$
\begin{aligned}
0 &= L'_{\lambda}(\omega, b, \lambda, \xi, \mu) = -(\sum_{n=1}^{N} t_n y(x_n) - 1 + \xi_n), \\
0 &= L'_{\omega}(\omega, b, \lambda, \xi, \mu) = \omega - \sum_{n=1}^{N} \lambda_n t_n \varphi(x_n) \Longrightarrow \omega = \sum_{n=1}^{N} \lambda_n t_n \varphi(x_n), \\
0 &= L'_{b}(\omega, b, \lambda, \xi, \mu) = -\sum_{n=1}^{N} \lambda_n t_n, \\
0 &= L'_{\xi_n}(\omega, b, \lambda, \xi, \mu) = C - \lambda_n - \mu_n \Longrightarrow \lambda_n = C - \mu_n.
\end{aligned}
\tag{19}
$$

We should have $\lambda_n \geq 0$ since they are Lagrange multipliers, and since $\mu_n \geq 0$ then $0 \leq \lambda_n \leq C$, $\sum_{n=1}^{N} \lambda_n t_n = 0$.

SVM: separation of overlapping classes

The parameter b can be determined from $t_n y(x_n) = 1$ for thus support vectors for which $0 < \lambda_n < C$ have $\xi_n = 0$:

$$1 = t_n y(x_n) = t_n (\sum_{m \in S} \lambda_m t_m k(x_n, x_m) + b). \tag{20}$$

The parameter b is then determined via averaging formula:

$$b = \frac{1}{N_M} \sum_{n \in M} \left(t_n - \sum_{m \in S} \lambda_m t_m k(x_n, x_m) \right) \tag{21}$$

M is set of all indices for which $0 < \lambda_n < C$.
Again, to classify new data points, we now determine the sign of

$$t_n y(x_n) = t_n(\omega^T \varphi(x_n) + b) = t_n (\sum_{n=1}^{N} \lambda_n t_n k(x, x_n) + b). \tag{22}$$