(67577) INTRODUCTION TO MACHINE LEARNING

Problem Set 9 – Kernels

Due: 1.6.2015 20:00

1 Kernel Ridge Regression

Let $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}$. And let $S = \{x_i, y_i\}_{i=1}^m \subseteq (\mathcal{X} \times \mathcal{Y})$ be a sample set. By applying the Tikhonov (ℓ_2) regularization to linear regression with the squared loss, we obtain the learning rule

$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \left(\frac{\lambda}{2} \|w\|^2 + \frac{1}{2m} \sum_{i=1}^{m} (\langle w, x_i \rangle - y_i)^2 \right) , \tag{1}$$

1. Find a closed form of the minimizer of Equation (1).
2. As in SVM, we can incorporate Kernels. Let ψ be a feature mapping from $\mathcal{X}$ into R^N. The corresponding RLM is

$$\underset{w \in \mathbb{R}^N}{\operatorname{argmin}} \left(\frac{\lambda}{2} \|w\|^2 + \frac{1}{2m} \sum_{i=1}^{m} (\langle w, \psi(x_i) \rangle - y_i)^2 \right) , \tag{2}$$

   Show how to implement the ridge regression algorithm with kernels.

2 Min-Kernel

Let N be any positive integer. For every $x, x' \in \{1, \dots, N\}$ define

$$K(x, x') = \min\{x, x'\}.$$

Prove that K is a valid Kernel, namely, find a mapping $\psi : \{1, \dots, N\} \to \mathbb{R}^d$ (for an appropriate value of d), such that

$$\forall x, x' \in \{1, \dots, N\},\ K(x, x') = \langle \psi(x), \psi(x') \rangle.$$

3 Practical Part - SVM

In this exercise you will implement the Soft-SVM algorithm with and without kernels. Download the file `ex9_code` from the course website. We provide you with two data sets, where the instances belong to $\mathbb{R}^2$ (thus, they can be visualized), and some auxiliary functions that will help you to visuzlize the prediction of SVM.

The exercise is divided into two parts. In the first part, you will implement linear SVM and use it to predict the labels of a data set which is approximately linearly separated. The second data set is far from being linearly separated. However, using the Gaussian kernel, we can still apply SVM (with kernels) to approximately find the correct labels.

3.1 Programming language

We wrote a school solution and some auxiliary functions using MATLAB[1]. However, you are allowed to use any programming language you prefer.

3.2 Linear SVM

1. Implement SGD for linear soft-SVM. The input of the algorithm is $(X, Y, lambda, T)$, where:

   (a) X is an $m \times d$ matrix, whose rows correspond to the instances.

   (b) Y is an $m \times 1$ matrix, where Y_i is the label of $X_{i,\cdot}$ (either 1 or -1).

   (c) $lambda$ is the regularization parameter.

   (d) T represents the number of iterations.

   The output, denoted w, is a $d \times 1$ vector, which is obtained by the soft-SVM algorithm.

2. Load `SVM_linear_data`, which contains a 2-dimensional input data `X,Y`. Run your algorithm with $T = 10m$, and $lambda = 0.01$ to get w, and then apply the function `show_SVM_linear(X,Y,w)` (where the m-file `show_SVM_linear` is provided by us[2]) to display the resulting classifier. Save the resulting plot as a JPEG file `SVM_linear.jpg`.

[1] We also provide a translation of this auxiliary functions to Python

[2] We also provide a translation of this file to Python, named `svm.utils`, which has been prepared by a student last year.

3. For debugging purposes, you can use the function `show_SVM_linear(X,Y,w)` inside each iteration of the algorithm, to show how the classifier is updated. Use the `pause()` command if the display runs too fast.

3.3 SVM with Gaussian Kernel

1. Implement SGD for soft-SVM with gaussian kernels (as described in lecture 8). The input of the algorithm is $(X, Y, lambda, sigma2, T)$, where:

   (a) X, Y, T, λ are defined as in the previous part (linear SVM).

   (b) $sigma2$ is the kernel width (e.g. σ^2 in $K(x, x') = \exp(-\|x - x'\|_2^2/\sigma^2)$

   The output, denoted $alphas$, is an $m \times 1$ vector, returned by the algorithm.

2. Important remarks:

   (a) Other than the for loop appearing explicitly in the pseudocode, you don't need any other loop in your code. This is important for your algorithm to run reasonably fast.

   (b) In particular, do not keep recalculating $K(x_i, x_j)$. Instead, compute (in advance) a matrix G of size $m \times m$, where $G_{i,j} = K(x_i, x_j)$, and use it throughout the run of the algorithm. Note that the calculation of G does not require loops. Here are some hints for this calculation:

      i. Let $Z = XX^\top \in \mathbb{R}^{m\times m}$. Note that $z_{i,j} = \langle x_i, x_j\rangle$.

      ii. Let $v \in \mathbb{R}^m$ be the diagonal of Z. Duplicate this vector m times. That is, use the function `repmat` (both in MATLAB and Python) to obtain a matrix $\bar{Z} \in \mathbb{R}^{m\times m}$ whose columns are equal to v. Denote the resulting matrix by D.

      iii. Express G in terms of Z and D.

   .

3. Load `SVM_gaussian_data`, which contains 2-dimensional input data `X,Y`.

4. Run your algorithm to get `alphas` as described next:

   (a) Set $T = 10m$ to be the number of iterations.

(b) Set `lambda`=0.1.

(c) Let `sigma2` vary over the following values: `sigma2`=10,`sigma2`=1, and `sigma2`=0.1.

(d) For each value of `sigma2` (and the corresponding output `alphas`), apply the function `show_SVM_gaussian(X,Y,alphas,sigma2)` (where the m-file `show_SVM_gaussian` is provided by us) to display the resulting classifier. Save the resulting plots as JPEG files `SVM_gaussian10.jpg`, `SVM_gaussian1.jpg`, and `SVM_gaussian01.jpg` respectively.

(e) Note: For debugging purposes only, you can use the function `show_SVM_gaussian(X,Y,w)` inside each iteration of the algorithm, to show how the classifier is updated. Use the `pause()` command if the display runs too fast.

3.4 Files Included in This Exercise

1. The data matrix X and the corresponding label vector Y are given in two alternative formats:

   (a) MATLAB format: `SVM_linear_data.mat`, `SVM_gaussian_data.mat`.

   (b) Text files: X_linear, Y_linear, X_gaussian, Y_gaussian.

2. `show_SVM_gaussian.m`, `show_SVM_linear.m`, `SVM_utils.py`

3.5 Submission

Upload to the course website a zip file named “ex9.zip” that contains the following files:

1. Your code.

2. The following figures: `SVM_linear.jpg`, `SVM_gaussian10.jpg`, `SVM_gaussian1.jpg`, and `SVM_gaussian01.jpg`.

3. A README file titled README with your username and ID number, separated by space. If you submit in pairs the README file should contain two lines, one for each of you. Here is an example for how the README file should look:
   mickey1 123456789
   minnie03 98765432

Hints

1. Section 1, first part:

   (a) Show that the RLM objective is convex.

   (b) Hence, we can find a solution by computing the derivative and comparing it to zero.

   (c) You may rely on the following fact: For every $d \times m$ matrix X and every $\beta > 0$, the matrix $XX^\top + \beta I$ is invertible.

2. Section 1, second part: The representer theorem tells us that there exists a vector $\alpha \in \mathbb{R}^m$ such that $\sum_{i=1}^m \alpha_i \psi(x_i)$ is a minimizer of Equation (2). This leads us to the following observations:

   (a) Let G be the Gram matrix with regard to S and K. That is, $G_{ij} = K(x_i, x_j)$. Note that G can be written as $X^\top X$ where X is a $N \times m$ matrix whose i'th column is $\psi(x_i)$. Define $g : \mathbb{R}^m \to \mathbb{R}$ by

   $$g(\alpha) = \frac{\lambda}{2}\alpha^T G\alpha + \frac{1}{2m}\sum_{i=1}^{m}(\langle \alpha, G_{\cdot,i}\rangle - y_i)^2 \ , \tag{3}$$

   where $G_{\cdot,i}$ is the i'th column of G. Show that if $\hat{\alpha}$ minimizes Equation (3) then $\hat{w} = \sum_{i=1}^m \hat{\alpha}_i \psi(x_i)$ is a minimizer of the RLM.

   (b) Show that g is convex. (You may use the fact that if $G = X^\top X$ for some matrix X then for every α it holds that $\alpha^\top G\alpha \geqslant 0$)[3]

   (c) Since G can be written as $X^\top X$ we get that $G + \beta I$ is invertible for every $\beta > 0$.

   (d) Find a closed form expression for $\hat{\alpha}$.

      i. Show that the gradient of the function $\alpha \mapsto \alpha^\top G\alpha$ is $2\alpha^\top G$.

      ii. For the second expression, note that $\frac{1}{2m}\sum_{i=1}^m(\langle \alpha, G_{\cdot,i}\rangle - y_i)^2$ can be viewed as a regression problem w.r.t. the G_i's.

[3] A matrix that satisfies one of these equivalent conditions is called a *"positive semidefinite matrix"*. See appendix C.3 in Shai & Shai