CSCI567 Fall 2013 - Assignment 4
Due date/time: Wed., Nov. 6, 2012, 3:15pm

October 22, 2013

Algorithm Component

1 Gaussian process

1. (10 points) In Gaussian process for regression, let us use linear kernel, namely, the covariance function is defined as $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j + \lambda \delta_{ij}$ where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise. Derive the resulting predictive distribution of predicting a single data instance $\boldsymbol{x}$. Also please contrast it to Bayesian linear regression. What are the similarity and difference?

2 Kernel methods

1. (2 points) $k(\boldsymbol{x}_m, \boldsymbol{x}_n) = (\boldsymbol{x}_m^{\mathrm{T}} \boldsymbol{x}_n + c)^d$ is a kernel function for $c \geq 0$ and d is a positive integer. That means, $k(\boldsymbol{x}_m, \boldsymbol{x}_n) = \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n)$ for some $\boldsymbol{\phi}(\boldsymbol{x})$.

Write down explicitly what this $\boldsymbol{\phi}(\boldsymbol{x})$ is — we have seen examples of it when we discussed $(\boldsymbol{x}_m^{\mathrm{T}} \boldsymbol{x}_n)^2$, which is a kernel function, in the class. You just have to generalize to our new kernel function.

2. (1 points) Prove that if $k_1(\boldsymbol{x}_m, \boldsymbol{x}_n)$ and $k_2(\boldsymbol{x}_m, \boldsymbol{x}_n)$ are kernel functions, then $\alpha k_1(\boldsymbol{x}_m, \boldsymbol{x}_n) + \beta k_2(\boldsymbol{x}_m, \boldsymbol{x}_n)$ is a kernel function if α and β are nonnegative.

3. (2 points) Prove that if $k_1(\boldsymbol{x}_m, \boldsymbol{x}_n)$ and $k_2(\boldsymbol{x}_m, \boldsymbol{x}_n)$ are kernel functions, then $k(\boldsymbol{x}_m, \boldsymbol{x}_n) = k_1(\boldsymbol{x}_m, \boldsymbol{x}_n) k_2(\boldsymbol{x}_m, \boldsymbol{x}_n)$ is a kernel function

4. (3 points) Consider a kernel function $k(\boldsymbol{x}_m, \boldsymbol{x}_n)$ and its associated mapping $\boldsymbol{\phi}(\boldsymbol{x})$, as well as the kernel matrix $\boldsymbol{K}$ computed over $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N$.

Consider a new mapping

$$\boldsymbol{\psi}(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x}) - \frac{1}{N} \sum_n \boldsymbol{\phi}(\boldsymbol{x}_n)$$

Namely, the mapping “centers" all the data such that $\frac{1}{N} \sum_n \boldsymbol{\psi}(\boldsymbol{x}_n) = \mathbf{0}$.

Now, let us use $\boldsymbol{\psi}(\boldsymbol{x})$ to compute the inner product matrix (i.e., Gram matrix) over the same $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N$. Call the matrix $\boldsymbol{G}$.

Prove that $\boldsymbol{G} = \boldsymbol{HKH}$ where $\boldsymbol{H}$ is a symmetrical matrix with rank of $(N-1)$. (This $\boldsymbol{H}$ matrix is called the centering matrix. $\boldsymbol{G}$ is called the centered kernel matrix of $\boldsymbol{K}$. We will use this type of matrix later .)

5. (3 points) This problem is to demonstrate the important of the matrix $\boldsymbol{H}$. In the lecture, we have determined that $d(\boldsymbol{x}_m, \boldsymbol{x}_n) = \|\boldsymbol{x}_m - \boldsymbol{x}_n\|_2^2$ is not a positive definite kernel. Nevertheless, let us construct a matrix $\boldsymbol{D}$ over $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N$. Let us make $\boldsymbol{D}$'s mn-th element be $d(\boldsymbol{x}_m, \boldsymbol{x}_n)$. Compute $\boldsymbol{G} = -\boldsymbol{HDH}$. What is $\boldsymbol{G}$? Is it positive definite? (The relationship between $\boldsymbol{G}$ and $\boldsymbol{D}$ was discovered by a mathematician in 1935 and this person went on delineating the properties of positive definite kernels, which formed the mathematical foundation of kernel methods in machine learning, ie, the material you have studied.)

3 Support vector machines

3.1 Support vector machines's dual formulation is convex (2 points)

The SVM's dual formulation is

$$\min_{\boldsymbol{\alpha}} \sum_n \alpha_n - \frac{1}{2} \sum_{mn} \alpha_m \alpha_n y_m y_n k(\boldsymbol{x}_m, \boldsymbol{x}_n)$$

with constraints $0 \le \alpha_n \le C$ and $\sum_n \alpha_n y_n = 0$.

Please verify the objective function is concave in terms of $\boldsymbol{\alpha}$. (Note that, maximizing a concave function is equivalent to minimizing a convex function, thus the optimization would be efficient to solve.)

3.2 Support vector machines using Gaussian kernels (4 points)

Consider a SVM classifier using the Gaussian kernel function

$$k(\boldsymbol{x}_m, \boldsymbol{x}_n) = \exp\{-\|\boldsymbol{x}_m - \boldsymbol{x}_n\|_2^2 / \sigma^2\}$$

where σ is called bandwidth. As in the Gaussian distribution, this bandwidth controls the shape of the kernel function — the smaller the bandwidth, the sharper the function looks. Obviously, our classifier's performance depends on the bandwidth.

How to choose a good bandwidth? Normally, this is done, in practice, with our good friend of cross-validation. However, in this problem, I am asking you to use your imagination to think about a special case, without doing experiments.

As before, assume we are given a training data set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i), i = 1, 2, \ldots, N\}$. Now, I am choosing one special $\sigma \ll 1$. Namely, the bandwidth is very small. In particular, I will choose the bandwidth to be

$$\sigma^2 = \min_{i \neq j} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 / 10$$

namely, the 1/10 of the smallest distance among all training data points[1]. My question is, *in light of such small a σ, how does the classifier behave?*

[1] 1/10 is a rather made-up number, I can choose 1/11, 1/100, etc

Specifically, for any new data point $\boldsymbol{x}$, the classifier computes the function

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i y_i k(\boldsymbol{x}, \boldsymbol{x}_i)$$

where α_i are Language multipliers. $\boldsymbol{x}$ is to be classified as 1 if $f(x)$ is positive or -1 if $f(x)$ is negative.

My intuition is that, with a small σ, the classification decision made by $f(x)$ is very much like a nearest neighbor classifier. **Present a clear and concise argument to show that my intuition is right**.

3.3 Defending Support Vector Machines (5 points)

If you have done the argument correctly in the previous section, you will now see why sometimes people call support vector machines "glorified nearest neighbor classifiers".

Suppose you are being interviewed for a job that requires machine learning skills. Your future manager does not believe fancy machine learning methods and has just commented exactly like "support vector machines are glorified nearest neighbor classifiers".

Now defend the benefits of support vector machines (SVM) over nearest neighbor classifications by listing a few points you think SVM is better than nearest neighbor classifiers.

You can also earn the full credit of this question by arguing (and convincing us) that nearest neighbor classifiers are better than SVMs. (You might find that direction might be substantially harder).

Note that, there is no ground-truth. As I mentioned, many machine learning algorithms have pros and cons. There is *not* a single algorithm that can do all. All you need to do here is to show a critical analysis of the algorithms.

3.4 Deriving Support Vector Machines without the Bias (2 points)

During the class, we had seen how SVM is formulated. Particularly, I have derived the SVM dual formulation using the labeling functions in the form of

$$y = \text{SIGN}(\boldsymbol{w}^\top \phi(\boldsymbol{x}) + b)$$

which results in a dual formulation with an equality constraint

$$\sum_n \alpha_n y_n = 0$$

where α_n is the Lagrange multiplier for the training data $(\boldsymbol{x}_n, y_n)$.

Now, consider the labeling function be

$$y = \text{SIGN}(\boldsymbol{w}^\top \phi(\boldsymbol{x}))$$

Note that there is no bias "b". Please derive the corresponding SVM formulation by following what we had done in the lecture (please use slack variables, assuming the data is not separable.)

4 Anomaly/outlier detection (10 points)

In this part of the assignment, you will be developing machine learning methods for the problem of detecting anomaly and outliers. In the programming component, you will be coding up the method you have developed and experimenting on data we have provided.

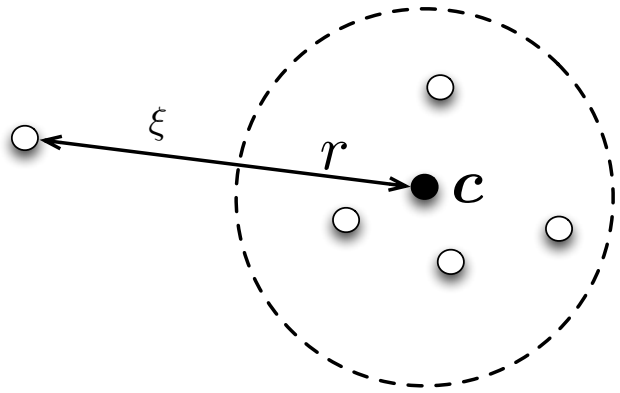

Figure 1: Plot for the problem of anomaly/outlier detection

The problem of detecting anomaly and outliers is of fundamental importance to many areas. For example, in network security, we can monitor the flow of information transmitted on the network. If the traffic patterns of the binary bits starts to change, we might conclude that there are abnormal activities on the network, for example, the network might be attacked.

While "normal" and "abnormal" seem to suggest a binary classification task, the problem in practice is not a classification. This is because from day to day, what is being considered as "normal" traffic patterns of bits on the network could be changing, depending on the behavior of the users of the network.

Thus, we will approach the problem in a different way. Assume we have received a lot of data $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\cdots$, and $\boldsymbol{x}_N$ as our training data. Note that there is no label associated with the data. Nonetheless, some of the data might be normal and some of the data might be abnormal.

Our approach builds on this intuition: normal data should look like each other and abnormal data should look different from normal data. Thus, to model "look like", we would like to say a normal data point $\boldsymbol{x}$ should be close to a center .

$$\|\boldsymbol{\phi}(\boldsymbol{x}) - \boldsymbol{c}\|_2^2 \leq r^2$$

where r is some number and $\boldsymbol{c}$ is the center — we we do not know their values yet. To model "look different", we would like to say an abnormal data point $\boldsymbol{x}$ should be far to the center of all data points.

$$\|\boldsymbol{\phi}(\boldsymbol{x}) - \boldsymbol{c}\|_2^2 \geq r^2$$

Graphically, our intuition is illustrated in Fig. 1. We draw a circle with radius of R around the center $\boldsymbol{c}$. All data points within the circle are regarded as "normal" and all outside "abnormal".

We tend to believe all normal data points should be able to form a small circle. Thus we are interested in finding a small r as much as possible. However, a small circle will make many data points becoming "abnormal". To balance these two conflicting forces, we minimize the following regularized objective function

$$\min_{r,\boldsymbol{c},\boldsymbol{\xi}} \quad \frac{1}{2}r^2 + C\sum_n \xi_n$$

subject to the constraint

$$\|\boldsymbol{\phi}(\boldsymbol{x}_n) - \boldsymbol{c}\|_2^2 \leq r^2 + \xi_n, \quad \forall\ n$$

where $\xi_n \geq 0$ is our slack variable ($\boldsymbol{\xi}$ includes all of them). A larger ξ_n implies that $\boldsymbol{x}_n$ is very far away from the center. C is our regularization parameter/coefficient. The larger the C is, the more unlikely we are to make data points as "abnormal". In other words, C controls the ratio of how many data points will become regarded as "normal".

The above formulation is in the primal form as we are using the transformed feature $\boldsymbol{\phi}(\boldsymbol{x})$. Note that the formulation looks like a SVM. **Please derive the corresponding dual formulation.**, just like we have done for SVM. Do not forget the constrain $\xi_n \geq 0$. You should:

1. Give the dual optimization problem in Lagrange multipliers. Your dual formulation will depend on inner products between transformed features. You should replace that with a kernel function $k(\cdot, \cdot)$.
2. You should also give the solution to the primal variables r and $\boldsymbol{c}$ in terms of these Lagrange multipliers.
3. Also write down the decision rule on how to determine whether $\boldsymbol{x}$ is normal or abnormal.

Note that all these should be given in terms of the kernel function, instead of the transformed features.

Programming Component (Anomaly Detection)

In Section 4 you were asked to derive the dual formulation of 'Anomaly detection'. A correct derivation should lead to a QP [2] that can be easily kernelized. Here we ask you to implement the Anomoly Detection algorithm by invoking Matlab's QP solver *quadprog* (*http://www.mathworks.com/help/optim/ug/quadprog.html*) and experimenting on a synthetic dataset.

5 Data

The file *'HW4-syn.mat'* (See figure 2) contains a synthetic dataset of 2D data points from two classes. The data is split into a train and test sets as usual. In our experiment, we will first treat class 1 (the blue ones) as the normal class to be modeled and class 2 (the red ones) as abnormal; then we reverse their roles.

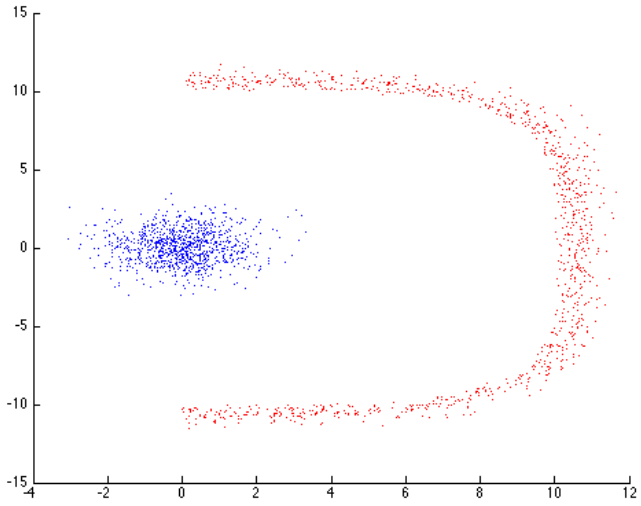

Figure 2: Figure for the Programming Component

6 Implementation

Using your derivation from Section 4, implement three functions:

1. *makePolyKernel* - a function that computes a **polynomial** kernel $K_{ij} = (x_i^T x_j + c)^d$.
2. *anomalyTrain* - Given the training set restricted to the jth class **only** ($j \in \{1, 2\}$) as well as parameters that define a polynomial kernel function, the function solves the dual optimization problem and return

[2] Quadratic Program - an optimization program with a quadratic objective function and linear equality or inequality constraints.

a model representing the jth class (Hint: represent the model using the support vectors and their dual parameters. Also compute the radius r).

3. *anomalyDetect* - Given a test set Z and the trained model of the jth class, this function determines for each data point in Z whether it belongs to the jth class (i.e., normal) or not (i.e., abnormal).

6.1 Experiment

For the experiment you should:

1. Using a fixed kernel function (see details in report) train a model that represents the training data points of class 1, but tune C such that about 5% of that training data is rejected. Then do the same for the training data points of class 2 (likely requiring a different C). Record the support vectors and their dual parameters.

2. Apply the two models on the **entire** test data which includes **both** class 1 **and** class 2. For each model, record the percentage of falsely rejected samples and falsely accepted samples.

   (a) For the jth model, where $j \in \{1, 2\}$, the falsely rejected samples are those samples from the jth class that our trained model rejected (false negatives).

   (b) Similarly, falsely accepted samples are those samples from **the other** class, that our trained model accepts (false positives).

Report

1. (25 points) Repeat the experiment defined in section 6.1 above for polynomial kernel with degree $d = 1, 2, 3$. Report per kernel and class $j \in \{1, 2\}$

   (a) The chosen C that provides roughly 5% false rejection and the resulting number of support vectors as well as the radius r

   (b) The false accept and false reject percentages.

2. (3 points) We will use Matlab's scatter() function to visually gain insights about the problem. First, plot the training data in 2D from both class 1 (in blue) and class 2 in red. This should give you the figure exactly the same as Fig. 2.

   Second, consider the model you have obtained that uses class 2 as training data (i.e., consider those data as "normal" data), mark the support vectors of the linear kernel model in black (circle them or plot on top of them using a big marker size) and for the quadratic model in magenta ('m').

3. (3 points) Given the plot, explain why the quadratic kernel outperforms the linear kernel on class 2.

Due Date and Time

The due date/time is **Nov. 6, 2013, 3:15pm**. A single two-day extension or two one-day extensions are allowed, combining with your previous requests. Other late submissions will get at most half the credit.

Collaboration

For the algorithm part of the assignment, you can collaborate. However, each has to turn in his/her own write-up.

For the coding part of the assignment, you may collaborate in groups of up to 4. Each group needs to submit to blackboard only once. Group members share the same credits.

Blackboard Submission Instructions

You are required to submit both runnable Matlab code files, result mats (.mat) and a formatted report (PDF) via the blackboard system.

Upload a single zip archive file *containing a single folder*. **The name of the zip file should be your abcd_hw4_AB.zip** where abcd is the last 4 digits of your USC ID and AB is your initials. **The folder should be named as results**. If you are working in a group, only one student from the group should upload the files. Please use the uploading student's ID number for the folder name.

The folder should contain all your .m files and one PDF report named *HW4Algorithm.pdf*. (for the algorithm part). Additionally, if you are submitting the coding part for the group, your folder should also contain *HW4Programming.pdf*. The report should contain the names and USC student IDs of all the group members. It should answer all programming questions — make you mark clearly each question you are answering. Since you will have plots, you will need to refer to the plots (such as "Please see Fig. 1" — LaTeX does not place plots next to texts so you need to number the plots and refer that way.)