10-715 Advanced Introduction to Machine Learning

Homework 2 *Due Oct 15, 10.30 am*

Rules

Please follow these guidelines. Failure to do so, **will** result in loss of credit.

1. Homework is due on the due date at 10.30 am. Please hand over your homework at the beginning of class. *Please see course website for policy on late submission.* **There is no grace period this time.**
2. We recommend that you typeset your homework using appropriate software such as LaTeX . If you are writing please make sure your homework is cleanly written up and legible. The TAs will not invest undue effort to decrypt bad handwriting.
3. You must hand in a hard copy of the homework. The only exception is if you are out of town in which case you can email your homeworks to *both* the TAs. If this is the case, your homeworks *must* be typeset using proper software. Please do *not* email written and scanned copies. Your email must reach the TAs by 10.30 am on the due date.
4. You are allowed to collaborate on the homework, but should write up your own solution and code. Please indicate your collaborators in your submission.
5. Please hand in the solutions to Problems 1,2 and Problem 3 separately. Write your name, andrew id and department on both submissions.
6. Please staple your homeworks.
7. If you are confused about of any of the terms you may refer Wikipedia. We have introduced some new concepts and methods that were not discussed in class. You should be able to find all of the required definitions on Wikipedia.

1 More Regression & Classification (Samy)

1.1 Optimal Classification & Regression

In class, we learned that "we cannot do better than the Bayes Classifier". In this problem we will try and formalize this notion.

In Classification and Regression our objective is to learn a rule $f : \mathcal{X} \to \mathcal{Y}$ which maps points from the space of inputs $\mathcal{X}$ to the space of outputs $\mathcal{Y}$. The loss function $l(x, y, f)$ for a pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$ measures how well $f(x)$ estimates y. The Risk $R(f)$ is the expectation of the loss under the joint distribution of (X, Y): $R(f) = \mathbb{E}_{XY}\left[l(X, Y, f)\right]$.

1. **(4 Points)** In Binary classification we use the loss $l(y, x) = \mathbb{1}(y \neq f(x))$. Then the classification Risk is $R(f) = \mathbb{E}_{XY}\left[\mathbb{1}(Y \neq f(X))\right] = \mathbb{P}_{XY}(Y \neq f(X))$. The Bayes Classifier is defined as $g(x) = \mathbb{1}(\mathbb{E}\left[Y|X = x\right] \geq 1/2)$. Show that for any f, $R(g) \leq R(f)$.
2. **(3 Points)** In Regression, if we use the squared error loss $l(x, y) = (y - f(x))^2$ the Risk is $R(f) = \mathbb{E}_{XY}\left[(Y - f(X))^2\right]$. Let $g(x) = \mathbb{E}\left[Y|X = x\right]$. Show that for any f, $R(g) \leq R(f)$.

1.2 Support Vector Regression

In this problem we study Support Vector Regression which works on the same premise as Support Vector Classification. We have training data $(x_i, y_i)_{i=1}^n$ and wish to find a function f such that $f(x) \approx y$.

For linear SV Regression we take $f(x) = w^\top x$. The most commonly used loss function in SV Regresion is the deadzone linear penalty $l_\epsilon(u) = \max(0, |u| - \epsilon)$. Accordingly, the SVR cost function is,

$$J(w) = \frac{1}{2}\|w\|^2 + c\sum_{i=1}^{n} l_\epsilon(w^\top x_i - y_i)$$

Here the $\frac{1}{2}\|w\|^2$ terms acts as a regularizer. It is usual to include an intercept term (i.e. $f(x) = w^\top x + b$) too but we will drop it here for simplicity.

1. **(4 Points)** By following essentially the same procedure as for SV Classification write down the dual problem as a QP.
2. **(1 Point)** Using the Dual Problem, develop a kernelized version of the algorithm. Also specify how you may obtain the prediction for a new point x_*.
3. **(1 Point)** How do you define "Support Vectors" for this problem ?
4. **Experiment (3 Points).** We will implement this on a 1D toy problem to gain some intuition. You need to write code to
   - Construct an RBF Kernel $K_{ij} = \exp(-\|x_i - x_j\|^2/2h^2)$.
   - Solve the Dual QP by taking in as arguments the Kernel Matrix, the labels, and the parameters c, ϵ. You should also return the support vectors. If you are using Matlab you could use CVX (`cvxr.com/cvx/`) or Matlab's inbuilt `quadprog` function.
   - Obtain the predictions of a new point taking in as inputs the inner products between the training and testing points and any of the dual variables.

   Please submit your code and the test results for linear and RBF Kernels on the given dataset. The test results should include a plot of the estimated function and identify the support vectors using a red cross. The given starter code does this for you so you can just attach the figures generated.

2 Expectation Maximization (Samy)

2.1 EM Basics

Prove the following statements about the EM Algorithm.

1. **(1 Point)** The expression we maximize at the M-step is a lower bound on the log likelihood.
2. **(1 Point)** The value of the Log-likelihood is nondecreasing at each iteration.

2.2 Pólya Mixture Model

The unsupervised version of Gaussian Discriminant Analysis is the Gaussian Mixture Model. In class we studied an algorithm for learning GMMs using EM. In this problem we will study an analogue for the Pólya Discriminant Analysis model from Homework 1 and develop a learning procedure in the EM framework.

2.2.1 Model (12 Points)

Our setting now is a document clustering problem into K groups using the Bag-of-Words model[1]. We have a vocabulary of V words. Each document is represented by $x \in \mathbb{N}^V$ which gives the number of times each word occurred in the document. The number of words $m = \sum_i x^{(i)}$ in a document differs from document to document. Refer Problem 2 in Homework 1 for other notation and definitions.

In our model, $z_i \in \{1, 2, \ldots, K\}$, $x_i \in \mathbb{N}^V$, for $i = 1, \ldots, n$. $\theta \in \Delta^{K-1}$, $\alpha_k \in \mathbb{R}^V$ for $k = 1, \ldots, K$. Our generative process is as follows. For each document in a corpus,

- $z \sim \text{Categ}\,(\theta)$
- $x_i | z_i = k, m_i$: First $\Delta^{V-1} \ni p \sim \text{Dir}\,(\alpha_k)$, then $x \sim \text{Mult}\,(m_i, p)$

As before, we will be Bayesian and assume a $\text{Dir}\,(\theta_0)$ prior for θ and $\mathcal{N}(0, \frac{1}{2\lambda} I_d)$ priors for all α_k's. Given data $(x_i)_{i=1}^n$ we wish to obtain the MAP estimates for $\theta, \alpha_1, \ldots, \alpha_K$. Develop an EM procedure to learn the parameters by maximizing the posteriors for the parameters. At the end of your derivations,

- Clearly state the E-step and the M-step.
- Give the prediction rule for a new point x_* obtained by maximizing the posterior probability of each class.

Note that if you are using Newton's method to optimize for α_k you can use a similar trick to homework 1 to perform the update in linear time.

2.2.2 Experiment (8 Points)

We will study how well this unsupervised version works on the same 2 datasets as before. You need to implement code to

1. Estimate $\theta, \alpha_1, \ldots, \alpha_K$.
2. Compute the predicted cluster/ group for a new point using the learned model.

Please submit your code. You should print it out and attach it to your solution.

For `data1.mat`, you should report the estimated parameters θ, $\alpha_1, \alpha_2, \alpha_3$, and the predictions for the given test set. For `data2.mat` you should report the estimated θ, the first 5 coordinates of $\alpha_1, \alpha_2, \alpha_3$, the predictions

[1] The purpose of learning a mixture model is not always to cluster the data but we will treat it as such here.

on the first 5 points in the test set and the accuracy on the test set. The given starter code does all of these for you so you can just attach screenshots of the results.

Note the following

- To initialize θ, α_k for EM, first cluster the data into K-clusters using K-means. Then use them as labels for the supervised Pólya Discriminant Analysis model from Homework 1. We have given a Matlab function that does this for you.
- To initialize α_k for Newton's method, set it to a reasonable point in Δ^{K-1}.
- Use $\lambda = 0.1$, $\theta_0 = (2, 2 \ldots, 2)^\top$.
- About 10 iterations of EM and 10 iterations of Newton's method should be sufficient.
- If you are using Matlab the following functions would be useful. `psi`, `gammaln`, `bsxfun`.
- Since Cluster Labels are arbitrary they will not correspond to the exact labels from the original training set. The given accuracy function takes this into consideration.

The following was the output of our implementation for `data1.mat`. If your answers do not correspond first double check with your peers and then speak to us.

```
>> q22

theta =
    0.3187
    0.3356
    0.3458


alpha =
    1.0846    0.8243    2.2359    3.5828
    1.0099    4.1057    0.9637    1.0056
    6.6758    1.5439    1.1636    1.2251


preds =
     2
     3
     2
     2
     1

Accuracy: 100.0000
>>
```

3 Kernels and RKHS (Veeru)

3.1 Image similarity functions

Let X be the space of arbitrary sized rectangular pictures where each pixel takes integer values from 0 to 255. Let a picture similarity function $k_1 : X \times X \to R$ be defined by

$$k_1(x, x') = \text{ number of 16x16 pixel patches common to } x, x'.$$

1. **(5 points)** Show that k_1 is a positive definite kernel.

2. **(5 points)** Show that the following similarity function is not a positive definite kernel.

$$k_2(x, x') = \begin{cases} 1 & \text{if } k_1(x, x') \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

   Hint: Show some points for which the Gram matrix is not positive semi-definite.

3.2 Positive definiteness of Gaussian Kernel

In this part, you will show that the Gaussian kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ defined by

$$k(x, x') = \exp\left(-\delta(\|x - x'\|^2), \delta > 0\right. \tag{1}$$

is a valid kernel.

1. **(2 points)** If k_1, k_2 are positive definite kernels, show that $\alpha k_1 + \beta k_2$ is a positive definite kernel for $\alpha, \beta \geq 0$.
2. **(5 points)** If k_1, k_2 are positive definite kernels, show that $k_1 k_2$ is a positive definite kernel. Hint: Consider the covariance matrices of multivariate Gaussian random variables U, V and their element-wise product.
3. **(3 points)** From the previous parts it is easy to see that if k is a positive definite kernel and $p : \mathbb{R} \to \mathbb{R}$ is a polynomial with positive coefficients, then $p \circ k$ is a positive definite kernel. Assuming this, show that $\exp(k)$ is a positive definite kernel.
4. **(5 points)** Show that the Gaussian kernel in (1) is a positive definite kernel. Hint: Write it as $\phi(x)\phi(x') \exp(\cdots)$. Please don't use the result from next subsection while showing this.
5. **(3 points)** Show this related result: if $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a positive definite kernel, then $\exp(-k)$ is not a positive definite kernel.

3.3 Checking validity by Fourier transforms

A function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ satisfying $k(x, y) = k'(x - y)$ for some $k' : \mathbb{R}^d \to \mathbb{R}$, is said to be a translation invariant function. Gaussian kernel is an example of such a function. There is a sometimes easy way to check if a translation invariant function is a positive definite kernel: If the Fourier transform of k' is non-negative, then k is a positive definite kernel. Using this fact, show that the following are positive definite kernels.

1. **(3 points)** The Gaussian kernel in (1)
2. **(3 points)**

$$k(x, y) = \frac{1}{1 + \|x - y\|^2}, k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$$

3.4 RKHS from the eigen functions of the kernel's integral operator

(5 points) We know by Mercer's theorem that if a $k : X \times X \to \mathbb{R}$ satisfies Mercer's condition, then $\forall x, x' \in X$, we can write

$$k(x, x') = \sum_{i=1}^{\infty} \lambda_i \phi_i(x) \phi_i(x')$$

where (ϕ_i) form an orthonormal basis of $L^2(X)$ and $\lambda_i > 0, \forall i \in \mathbb{N}$. Let

$$\mathcal{H} = \left\{ \sum_{j=1}^{\infty} a_j \phi_j \mid \sum_{j=1}^{\infty} a_j^2/\lambda_j < \infty \right\}.$$

$\mathcal{H}$ is a Hilbert space, when equipped with the inner product

$$\left\langle \sum_{j=1}^{\infty} a_j \phi_j, \sum_{j=1}^{\infty} b_j \phi_j \right\rangle_{\mathcal{H}} = \sum_{j=1}^{\infty} \frac{a_j b_j}{\lambda_j}.$$

Show that the reproducing property holds, that is, for $f \in \mathcal{H}, x \in X$, show that

$$\langle f, k(., x) \rangle_{\mathcal{H}} = f(x).$$

3.5 Optimizing over an RKHS

(5 points) Let $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a positive definite kernel and $\mathcal{H}_k$ be the RKHS for which k is the reproducing kernel. Suppose $(x_i, y_i)_{i=1}^n$ are data points in $\mathbb{R}^d \times \mathbb{R}$ and we want to find a function $\widehat{f} \in \mathcal{H}_k$ such that $\widehat{f}(x_i)$ approximates y_i. Solving the following regularized problem is a natural way of doing this.

$$\min_{f \in \mathcal{H}_k} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \|f\|_{\mathcal{H}_k}^2 \tag{2}$$

Here $\lambda > 0$ is a regularization parameter. This is typically an infinite-dimensional problem. Convert this into a finite dimensional problem, solve it and show that the fitted values satisfy $\widehat{y} = Sy$ for some matrix S where $y \in \mathbb{R}^n$ is the concatenation of y_i's.

3.6 Some computational considerations for SVM

Suppose we have m data points $(x_i, y_i)_{i=1}^m$ from $\mathbb{R}^d \times \{-1, +1\}$ and we would like to train a soft-margin kernel SVM. In many practical scenarios, m is very large(say hundreds of millions).

1. **(1 points)** If we implement kernel SVM naively, what is the space complexity? Give the main terms assuming $m \gg d$.
2. **(2 points)** What is the cost of computing the decision function at a given x, assuming that there are $O(m)$ support vectors and $k(x, x')$ takes $O(d)$ time? This can be quite slow when m is very large.
3. **(3 points)** There are some ways to approximate the decision function. Read the introduction of [Le et al., 2013] and write down the computational complexity of decision function in their setting.

References

[Le et al., 2013] Le, Q., Sarlos, T., and Smola, A. (2013). Fastfood - approximating kernel expansions in loglinear time. In *30th International Conference on Machine Learning (ICML)*.