FINAL PRACTICE PROBLEMS # 2

Sunday, 10th June 2018

Problem 1 (Kernel k-means)

Suppose we have a dataset $\{x_i\}_{i=1}^n, x_i \in \mathbb{R}^\ell$ that we want to split into k clusters, i.e., finding the best k-means clustering. Furthermore, suppose we know *a priori* that this data is best clustered in an impractically high-dimensional feature space $\mathbb{R}^m$ with an appropriate metric. Fortunately, instead of having to deal with the (implicit) feature map $\phi : \mathbb{R}^\ell \to \mathbb{R}^m$ and (implicit) distance metric[1], we have a kernel function $K(x_1, x_2) = \phi(x_1)^T \phi(x_2)$ that we can compute easily on the raw samples. How should we perform the kernelized counterpart of k-means clustering?

Derive the underlined portion of this algorithm.

Algorithm 1 Kernel K-means

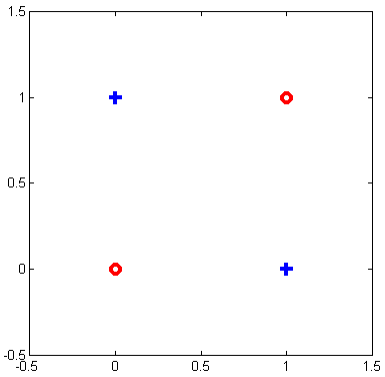
Require: Data matrix $X \in \mathbb{R}^{n \times \ell}$; Number of clusters k; kernel function $\kappa(x_1, x_2)$
Ensure: Cluster id $i(j)$ for each sample x_j.

```
function KERNEL-K-MEANS(X, k)
    Randomly initialize i(j) to be an integer in 1, 2, ..., k for each x_j.
    while not converged do
        for i ← 1 to k do
            Set C_i = {j ∈ {1, 2, ..., n} : i(j) = i}.
        end for
        for j ← 1 to n do
            Set i(j) = argmin_i _____
        end for
    end while
    Return C_i for i = 1, 2, ..., k.
end function=0
```

(Hint: there will be no explicit representation of the "means" $\vec{\mu}_i$, instead each cluster's membership itself will implicitly define the relevant quantity, in keeping with the general spirit of kernelization that we've seen elsewhere as well.)

[1] Just as how the interpretation of kernels in kernelized ridge regression involves an implicit prior/regularizer as well as an implicit feature space, we can think of kernels as generally inducing an implicit distance metric as well. Think of how you would represent the squared distance between two points in terms of pairwise inner products and operations on them.

Problem 2 (Boosting)



Prove that boosting with decision stumps cannot have zero error in training set which resembles XOR function?

Problem 3 (SVM)

Recall the soft-margin SVM in the primal:

$$\begin{aligned} \arg\min_{\mathbf{w},b,\{\xi_n\}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n & \\ y_n(\mathbf{w}^T\mathbf{x}_n + b) &\geq 1-\xi_n \quad n \in \{1,\ldots,N\} \\ \xi_n &\geq 0 \quad n \in \{1,\ldots,N\} \end{aligned}$$

(a) Suppose you are given the solution to this problem but only for $(\mathbf{w}^*, b^*)$. Instances 1 and 2 are the support vectors. Compute the optimal values of the slack variables from these values.

(b) What is the effect of increasing C on the following quantities?

i. The margin

ii. The number of support vectors

Problem 4 (Kernel)

The polynomial kernel is defined to be

$$k(\mathbf{x},\mathbf{y}) = (\mathbf{x}^T\mathbf{y} + c)^d$$

Where $\mathbf{x},\mathbf{y} \in \mathbb{R}^n$, and $c \geq 0$. When we take $d = 2$, this kernel is called the quadratic kernel. Find the feature mapping $\Phi(\mathbf{z})$ that corresponds to the quadratic kernel.

Problem 5 (Logistic Regression and Perceptron **(28 pts)**)

Given data $\mathcal{D} : \{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \ldots, (\boldsymbol{x_n}, y_n)\}$ and the query point $\boldsymbol{x}$ we would like to choose $\boldsymbol{w}$ that minimizes the loss which is the negative log likelihood.

$$J(\boldsymbol{w}) = -\sum_{i=1}^{n} [y_i \log h_{\boldsymbol{w}}(\boldsymbol{x}_i) + (1 - y_i) \log(1 - h_{\boldsymbol{w}}(\boldsymbol{x}_i))]$$

where $h_{\boldsymbol{w}}(\boldsymbol{x}_i) = \frac{1}{1+\exp(-\mathbf{w}^T \boldsymbol{x}_i)}$.
The gradient descent update rule as we have seen in class is

$$\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t - \eta \sum_{i=1}^{n} (h_{\boldsymbol{w}^t}(\boldsymbol{x}_i) - y_i)\boldsymbol{x}_i$$

(a) Find the Hessian of $J(\boldsymbol{w})$ and show that it is positive semidefinite.

(b) Write the stochastic gradient descent update rule for logistic regression.

(c) Moving from soft to hard decision regions, replace $h_{\boldsymbol{w}^t}(\mathbf{x}_i)$ in the SGD update rule with $\hat{y}_i^t$ where

$$\begin{aligned} \hat{y}_i^t &= 1 \quad \text{if} \quad h_{\boldsymbol{w}^t}(\mathbf{x}_i) \geq 0.5 \quad \text{or} \quad \boldsymbol{w}^{tT} \boldsymbol{x}_i \geq 0 \\ &= 0 \quad \text{if} \quad h_{\boldsymbol{w}^t}(\mathbf{x}_i) < 0.5 \quad \text{or} \quad \boldsymbol{w}^{tT} \boldsymbol{x}_i < 0 \end{aligned}$$

Simplify to show that this is akin to the Perceptron update rule.

Problem 6 (EM algorithm and Gaussian Mixture Model)

Suppose that we are fitting a Gaussian mixture model for data items consisting of a single real value, x, using $K = 2$ components. We have $N = 5$ training cases, in which the values of x are as follows:

$$5, 15, 25, 30, 40$$

We use the EM algorithm to find the maximum likelihood estimates for the model parameters, which are the mixing proportions for the two components, w_1 and w_2, and the means for the two components, w_1 and w_2. The standard deviations for the two components are fixed at 10.

Suppose that at some point in the EM algorithm, the E step found that the posterior probabilities of the two components for the five data items were as follows:

r_{i1}	0.2	0.2	0.8	0.9	0.9
r_{i2}	0.8	0.8	0.2	0.1	0.1

What values for the parameters w_1, w_2, μ_1 and μ_2 will be found in the next M step of the algorithm?