Machine Learning 10-701 February 6, 2013
http://alex.smola.org/teaching/cmu2013-10-701/ due on February 20, 2013
Carnegie Mellon University

Homework 2

Instructions

- The homework is due in the lecture on February 20. Anything that is received after the lecture will not be considered.
- Please submit one set of notes for each of the problems and put them into a separate stack. Don't forget to add your name on each sheet.
- Alternatively, you can e-mail your solution to `10.701.homework@gmail.com`. Please write the subject as: "[Homework 2] *yourandrewID*", followed by the numbers of the questions you are including in the email, or "all" if you're submitting the entire homework. As before, the cutoff is the end of the lecture.
- If you are on the waitlist, please either e-mail your solution or hand it in with everyone else. However, put it into an envelope and write waitlist on it. While we cannot guarantee that you will definitely get a spot, we will give preference to students who submitted homework.
- If you submit code, it should be sufficiently well documented that the TAs can understand what is happening. Also attach pseudocode if you feel that this makes the result more comprehensible.

1 Lagrange multipliers in entropy maximization (Xuezhi)

A **discrete** distribution $p = (p_1, p_2, \ldots, p_n)$ has $\sum p_i = 1$ and $p_i \geq 0$ for all i. The entropy, which measures the uncertainty of a distribution, is defined by $H(p) = -\sum_{i=1}^{n} p_i \log p_i$. (Note we define $0 \log 0 = 0$).

1. Show that the uniform distribution has the largest entropy.
   [Hint: Use Lagrange multipliers:
   http://en.wikipedia.org/wiki/Lagrange_multipliers]

The entropy of a **continuous** distribution with density function f is defined by $H(f) = -\int f(x) \log f(x)\, dx$.

Questions 2 and 3 are extra credit:

Let X be a random variable with density f.

2. Prove that if $\mathbb{E}_f[X] = 0\, \mathbb{E}_f[X^2] = \sigma^2$, then the Gaussian distribution $N(0, \sigma^2)$ has the maximal entropy.
   [Hint: Use Lagrange multipliers, and use $\frac{\partial}{\partial f(y)} \int r(x) f(x)\, dx \;=\; r(y)$*. To prove, let* $F[f(x)] \doteq \int r(x) f(x) dx$ *functional. We have to calculate* $\frac{\partial}{\partial f(y)} F[f(x)]$*. By defintion*

   $$\frac{\partial}{\partial f(y)} F[f(x)] \doteq \lim_{\epsilon \to 0} \frac{F[f(x) + \epsilon \delta(x-y)] - F[f(x)]}{\epsilon},$$

   where $\delta(x)$ *is the Dirac delta.*
   http://en.wikipedia.org/wiki/Functional_derivative

   $$\begin{aligned} \lim_{\epsilon \to 0} \frac{F[f(x) + \epsilon \delta(x-y)] - F[f(x)]}{\epsilon} &= \frac{\int r(x)(f(x) + \epsilon \delta(x-y))\, dx - \int r(x) f(x)\, dx}{\epsilon} \\ &= \frac{\int \epsilon r(x) \delta(x-y)\, dx}{\epsilon} \\ &= r(y) \end{aligned}$$

   Similarly, one can also prove that $\frac{\partial}{\partial f(y)} \int f(x) \log f(x)\, dx \;= \log f(y) + 1$
   Also, if a density has the form $a \exp((x-b)^2/2c^2)$ *for any real constants* a*,* b *and* c*, then it must be the density*

of the normal distribution.]

3. Prove that if $supp(f) = [0, \infty]$, and $E[X] = \mu$, then the exponential distribution ($f(x) = \frac{1}{\mu}\exp(-\frac{x}{\mu})$) has the largest entropy.
   [Hint: If a density has the form $a\exp(bx)$ for any real constants a, b, then it must be the density of the exponential distribution.]

2 The Perceptron is NOT Limited (Mu)

Given a sequence of n samples $x_i \in \mathbb{R}^p$, and the according labels $y_i \in \{-1, 1\}$, the Perceptron algorithm runs as follows:

```
initialize w ← 0
for i = 1, ..., n do
  if y_i⟨w, x_i⟩ < 0 then
    w ← w + y_i x_i
  end if
end for
```

For simplicity, we have put the bias b into w, that is $w \leftarrow [w, b]$ and $x_i \leftarrow [x_i, 1]$. So there is no necessary to consider the bias term on this question.

Assume $\|x_i\| \le r$ for all i. Alex has shown that if there exists some w^* such that $\|w^*\| = 1$ and for all i

$$y_i\langle w^*, x_i\rangle \ge \rho,$$

then the number of updates of w is upper bounded by

$$r^2/\rho^2. \tag{1}$$

1. Consider the general Perceptron updates $w \leftarrow w + \eta y_i x_i$ with learning rate $\eta > 0$. The Perceptron algorithm is the special case $\eta = 1$. Prove a bound on the number of updates similar to (1). How does η affect this bound?

2. From the bound (1), we know small ρ may make the problem hard to solve. Actually, it could be exponentially hard! Consider the following example:

$$y_i = (-1)^{i+1} \text{ and } x_i = (\underbrace{(-1)^i, \ldots, (-1)^i, (-1)^{i+1}}_{i \text{ elements}}, 0, \ldots, 0) \text{ for } i = 1, \ldots, m$$

   Prove that $O(2^m)$ updates are required before the Perceptron algorithm finds an optimal w^* that satisfies $y_i\langle x_i, w^*\rangle > 0$ for all i, no matter how we select the sample on each iteration. (That is, no matter how you construct the sequence of samples, it should be at least $O(2^m)$ length so that the Perceptron algorithm can find the optimal solution.)

3. Now let's drop the assumption that samples are linear separable. The world is not so simple! But fortunately, this world is not so complex.

   Let u be any vector with $\|u\| = 1$ and let $\rho > 0$. Define the deviation of x_i by $d_i = \max(0, \rho - y_i\langle u, x_i\rangle)$, and $\delta = \left(\sum_{i=1}^n d_i^2\right)^{-1/2}$. Show that the number of updates of the Perceptron algorithm is bounded by $(r + \delta)^2/\rho^2$.

[Hint: You may try to construct separable x'_i and reuse the previews proof. For example, define $x'_i = [x_i, 0, \ldots, 0, c, 0, \ldots, 0]$, which is a $p+n$ dimensional vector and the scalar c is on position $p+i$. Then how to construct the according u' such that u' separates x'_is? Now you get a bound similar to (1), then how to choose the scale c to minimize this bound? Does the bound you obtained work on the original x_i?]

4. The main idea behind the proof of the above question is mapping sample points into higher dimensional space where linear separation is possible. It coincides with the idea of kernel methods, which mapping x_i into $\phi(x_i)$.

   Write down the dual Kernel Perceptron algorithm. Your inputs are a kernel function $k(\cdot,\cdot)$ and the sample sequence $\{y_i, x_i\}_{i=1}^n$ similar as above. Your output will be the coefficients $\{\alpha_i\}_{i=1}^n$, where $w = \sum_{i=1}^n \alpha_i x_i$. Also show how to predict ($y = 1$ or -1) when a new sample x is coming.

   [Hint: Examine the Perception algorithm again, you will find that w is a linear combination of x_i, denoted by $w = \sum_{i=1}^n \alpha_i x_i$. So we only need to get the values of α_is. Instead of updating w, the dual Perceptron updates $(\alpha_1, \ldots, \alpha_n)$ directly. Then replace x_i with $\phi(x_i)$. Note that, we only need to compute $\langle \phi(x_i), \phi(x_j) \rangle$ during updating, which could be presented as the kernel function $k(x_i, x_j)$.]

3 Linear Algebra (Leila)

3.1 Elementwise product of two positive semidefinite matrices

Let $K_1, K_2 \in \mathbb{R}^{n\times n}$ be two positive semidefinite matrices. Prove that their **elementwise** product matrix $K(i,j) = K_1(i,j)K_2(i,j)$ is positive semidefinite matrix, too.

[Hint: Consider the covariance matrix of $w = (u_1v_1, \ldots, u_nv_n)^T$ vector, where the n dimensional vectors $u = (u_1, \ldots, u_n)^T \sim N(0, K_1)$ and $v = (v_1, \ldots, v_n)^T \sim N(0, K_2)$ are each drawn from its own Gaussian distribution, as shown here. Remember that the covariance matrix is always positive semidefinite.]

3.2 Product of positive semidefinite matrices

Let $A, B \in \mathbb{R}^{n\times n}$ be positive semidefinite matrices.

1. Show that AB is not necessarily positive semidefinite.

2. Show that A^m is positive semidefinite for all $m \in \mathbb{Z}_+$.
   [Hint: The finite-dimensional spectral theorem says that any symmetric matrix $M \in \mathbb{R}^{n\times n}$ can be diagonalized by an orthogonal matrix. More explicitly: For every symmetric real matrix M there exists a real orthogonal matrix U such that $D = U^TMU \in \mathbb{R}^{n\times n}$ is a diagonal matrix.
   (Orthogonal means $U^TU = I$ where I is the identity matrix.).]

4 Kernels (Ina)

4.1 Constructing kernels

Let $k_1(x, \tilde{x})$ and $k_2(x, \tilde{x})$ be valid kernel functions, and $c_1, c_2 > 0$ be positive real constants.

1. Show that $c_1k_1(x, \tilde{x}) + c_2k_2(x, \tilde{x})$ is a valid kernel function too.

2. Show that $k_1(x, \tilde{x})k_2(x, \tilde{x})$ is also a valid kernel function.
[Hint: Use the previous results about the elementwise products of positive definite matrices. Remember that if k is a kernel, then any Gram matrix made of $k(\cdot, \cdot)$ is positive semi definite.]

4.2 Non-kernels

1. Let $k_1(x, \tilde{x})$ and $k_2(x, \tilde{x})$ be valid kernel functions. Show that $k_1 - k_2$ is not necessarily positive semi definite.
[Hint: Remember that if k is a kernel, then any Gram matrix made of $k(\cdot, \cdot)$ is positive semi definite.]

2. We know that $\exp(-\|x - y\|^2)$ is a kernel function. Show that $\exp(\|x - y\|^2)$ is not a valid kernel.
[Hint: Construct a Gram matrix that is not positive semi definite.]

4.3 Representer theorem

Let $\mathcal{F}$ be an RKHS function space with kernel $k(\cdot, \cdot)$. Let $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ be m training input-output pairs. Our task is to find the $f^* \in \mathcal{F}$ function that minimizes the following regularized functional:

$$f^* = \arg\min_{f \in \mathcal{F}} \left(\prod_{i=1}^{m} |f(\mathbf{x}_i)|^6 \right) \sum_{i=1}^{m} \left[\left| \sin\left(\|\mathbf{x}_i\|^{|y_i - f(\mathbf{x}_i)|} \right) \right|^{25} + y_i |f(x_i)|^{42} \right] + \exp(\|f\|_{\mathcal{F}})$$

This is a nonparametric minimization problem over functions in the function space $\mathcal{F}$. Prove that f^* can be expressed as $f^*(\cdot) = \sum_{i=1}^{m} \alpha_i k(x_i, \cdot)$, reducing the problem to an m-dimensional minimization [with respect to $(\alpha_1, \ldots, \alpha_m)$] only.
[Hint: Use the representer theorem; see http://en.wikipedia.org/wiki/Representer_theorem]

5 SVMs (Junier)

Implement the soft SVM classification problem in Primal and Dual form. (**It is allowed to use any quadratic programming codes and toolboxes (e.g. 'quadprog' in Matlab), but do NOT use SVM toolboxes.**)

5.1 Primal Problem

Implement a function predictedY = svm_primal_classify(testX, trainX, trainY, C) that solves the primal problem for SVMs using trainX, trainY as training data, labels (respectively), and C as the constant in the primal formulation, then using testX as test data, the function outputs predictedY, the predicted labels.

[Hint: The primal problem is :

$$\min_{w} \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{m} \xi_i$$

subject to

$$y_i \langle x_i, w \rangle \geq 1 - \xi, (i = 1, \ldots, m)$$
$$\xi \geq 0, (i = 1, \ldots, m)$$

]

5.2 Dual Problem

Implement a function predictedY = svm_dual_classify(testX, trainX, trainY, C, kernel) that solves the dual problem for SVMs using trainX, trainY as training data, labels (respectively), C as the constant in the dual formulation, and kernel $\in$ {'linear',' polynomial',' RBF'} is a string specifying the kernel to use (see below) then using testX as test data, the function outputs predictedY, the predicted labels.

For kernel = 'linear', let $K(x_i, x_j) \equiv \langle x_i, x_j \rangle$, for kernel = 'polynomial', let $K(x_i, x_j) \equiv (\frac{1}{2}\langle x_i, x_j \rangle)^3$ and for kernel = 'RBF', let $K(x_i, x_j) \equiv \exp(-\frac{1}{2}\|x_i - x_j\|_2^2)$.

[Hint: The dual problem is :

$$\begin{array}{cc} \max_{\alpha} & -\frac{1}{2}\sum_{i,j}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) + \sum_i^n \alpha_i \\ \textit{subject to} & 0 \leq \alpha_i \leq C \\ & \sum_{i=1}^{n} \alpha_i y_i = 0 \end{array}$$

]

5.3 Results on Data

Compare the classification accuracy on the training set with 'linear' 'polynomial', and 'RBF' kernels using both svm_primal_classify and svm_dual_classify. (Note: RBF kernel need only be tried in dual formulation. Why is this?) Use the data.mat dataset in http://alex.smola.org/teaching/cmu2013-10-701/assignments/data.mat (first two columns are features, third column is binary label). Also, provide decision surface plots.

[Hint: Consider using 'contourf' and 'meshgrid' MATLAB functions]