10-715 Advanced Introduction to Machine Learning

Homework 1 *Due Oct 1, 10.30 am*

Rules

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.*
2. You have a grace period until 3pm on the same day to submit the homework. If you cannot submit it in class, please hand it over to the TAs (or slide it under their office doors) with "10715 - Homework 1" clearly written on the front page.
3. 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.
4. 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 3pm on the due date.
5. You are allowed to collaborate on the homework, but should write up your own solution and code. Please indicate your collaborators in your submission.
6. Please hand in the solutions to Problems 1,2 and Problems 3, 4 separately. Write your name, andrew id and department on both submissions.
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 Regression (Samy)

1.1 Multi-Task Regresion

In class we considered the Linear regression problem $y = \beta^\top x$ where $x \in \mathbb{R}^d, \beta \in \mathbb{R}^d$ and $y \in \mathbb{R}$. Say we now want to consider predicting a vector of outputs $\mathbf{y} \in \mathbb{R}^k$ via a prediction matrix $\Theta \in \mathbb{R}^{k \times d}$.

We have training data $(x_i, y_i)_{i=1}^n$ where $x_i \in \mathbb{R}^d, y_i \in \mathbb{R}^k$. Let $\mathbf{Y} \in \mathbb{R}^{k \times n}$ denote a matrix with y_i in each column and $X \in \mathbb{R}^{d \times n}$ with x_i in each column. We wish to obtain $\widehat{\Theta}$ such that $\mathbf{Y} \approx \widehat{\Theta} X$. The way we have set it up, each row corresponds to a "task".

1. **(1 Point)** If the cost function is $J_0(\Theta) = \|\mathbf{Y} - \Theta X\|_F^2$, show that the solution to the combined problem can be obtained via the solutions to each individual task. Here, $\|\cdot\|_F$ denotes the Frobenius norm.
2. **(6 Points)** Now we use the cost function $J(\Theta) = J_0(\Theta) + \lambda \mathcal{C}(\Theta)$ where $\mathcal{C}(\cdot)$ is a complexity penalty. For each of the following $\mathcal{C}(\cdot)$ indicate whether the tasks are still (a)independent, (b) convex. (c) Explain why such a penalty function would be useful.
   (a) $\mathcal{C}(\Theta) = \|\Theta\|_F^2$
   (b) $\mathcal{C}(\Theta) = \sum_{j=1}^d \sqrt{\sum_{i=1}^k \Theta_{ij}^2}$
   (c) $\mathcal{C}(\Theta) = \text{rank}(\Theta)$

1.2 Shrinkage in Ridge Regression

We will study some of the properties of the Ridge Regression Problem

$$\widehat{\beta} = \underset{\beta}{\text{argmin}} \|\mathbf{y} - X\beta\|_2^2 + \lambda \|\beta\|^2$$

where $X = [X_1^\top; \ldots; X_n^\top] \in \mathbb{R}^{n \times d}$, $\beta \in \mathbb{R}^d$, $\mathbf{y} \in \mathbb{R}^n$. Assume that $\mathbb{E} X_i = \mathbf{0}$ and $\text{rank}(X) = d$. Let the SVD of X be $X = U \Sigma V^\top$ where $U = [u_1, \ldots, u_d] \in \mathbb{R}^{n \times d}$, $V = [v_1, \ldots, v_d] \in \mathbb{R}^{d \times d}$ and $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_d) \in \mathbb{R}^{d \times d}$.

1. **(3 Points)** Show that the prediction at a new point $\mathbb{R}^d \ni x_* = V z_*$ can be written as,

$$x_*^\top \widehat{\beta} = \sum_{i=1}^d \frac{z_{*i} \sigma_i}{\sigma_i^2 + \lambda} u_i^\top y$$

2. **(3 Points)** Use the above expression to explain how Ridge regression helps in controlling the variance in a linear model.

1.3 Local/Weighted Linear Regression

1. **(2 points)** In class we looked at locally weighted linear regression where we predicted $\hat{f}(x) = \widehat{\beta}^\top x$ where

$$\widehat{\beta} := \underset{\beta}{\text{argmin}} \sum_{i=1}^n w_i(x)(y_i - \beta^\top x_i)^2$$

   A typical choice for $w_i(x)$ is $k((x - x_i)/h)$ where k is a smoothing kernel–hence the name local linear regression. Show that

$$\hat{f}(x) = x^\top (X^\top W X)^{-1} X^\top W \mathbf{y}$$

   where $X \in \mathbb{R}^{n \times d}$ is the data matrix, $W = \text{diag}(w_1(x), w_2(x), \ldots, w_n(x)) \in \mathbb{R}^{n \times n}$ and $\mathbf{y} \in \mathbb{R}^n$ is a vector of training labels.

N.B: Locally Polynomial Regression refers to essentially the same idea but now you augment your X with higher order polynomial terms of the features in addition to the constant and linear terms.

2. **(1 + 1 points)** If instead we solve for a locally "constant" estimate with the choice of w_i as given above,

$$\hat{f}(x) := \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{n} w_i(x)(y_i - \theta)^2$$

show that we obtain the Nadaraya Watson Estimator. Argue that $\min_i y_i < \hat{f}(x) < \max_i y_i$, $\forall x$.

1.4 Least Norm Solution

1. **(3 points)** We saw that the least squares solution to the overdetermined problem $\mathbf{y} \approx X\beta$ was $\widehat{\beta} = (X^\top X)^{-1} X^\top \mathbf{y}$. Here, $X \in \mathbb{R}^{n \times d}$, $\mathbf{y} \in \mathbb{R}^n$, $\beta \in \mathbb{R}^d$ and $n > d$. If the system is underdetermined ($n < d$), one technique is to add an ℓ_1 or ℓ_2 penalty. Alternatively, we could find the point with the lowest norm $\|\beta\|$ in the affine subspace that solves $y = X\beta$. By assuming that X is full rank, find the least norm solution.

2. **(5 points)** Now the system is still under determined but X is also not full rank. Hence we are not guaranteed to be able to solve $\mathbf{y} = X\beta$ exactly. We wish to find the point with the lowest norm $\|\beta\|$ in the affince subspace that minimizes $\|\mathbf{y} - X\beta\|$. Show that this "least-squares"–"least-norm" solution is given by $\hat{\beta} = X^\dagger \mathbf{y}$ where $X^\dagger$ is the Moore-Penrose inverse of X.

N.B: Please do not use part 2 as a starting point for part 1. We expect you to solve part 1 from scratch. Once you've done part 2, you will see that both the least squares solution and the least norm solution can be expressed this way.

2 Pólya Discriminant Analysis (Samy)

In class we studied Generative models for Classification – Gaussian Discriminant Analysis which applies to continuous data is a canonical example. Here, we will look a generative classification model for discrete data such as text corpora. **This is not a popular method.** (At least not yet !) But it will help you gain some intuition on parametric probabilistic modeling and Maximum Likelihood Estimation in a fairly complex model.

We first introduce some notation. Categ , Mult , Dir etc refer to the Categorical, Multinomial and Dirichlet Distributions respectively. There is confusion on the definitions of the Categorical and Multinomial distributions. Here we treat them as the analogues of the Bernoulli and Binomial distributions in the binary case. (You may refer Wikipedia for proper definitions.) Δ^{N-1} refers to the $(N-1)-$simplex: $q \in \Delta^{N-1} \subset \mathbb{R}^N \implies q^{(i)} > 0, \sum_i q^{(i)} = 1$. We will use subscripts to index documents/labels etc in the corpus (e.g. x_i, y_i) and superscripts to index elements of the vector (e.g. $x^{(i)}$).

The setting is a document classification problem for K classes using the Bag-of-Words model. 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. In addition, for each document we have the class label $y \in \{1, 2, \ldots, K\}$.

2.1 Model

$y_i \in \{1, 2, \ldots, K\}$, $x_i \in \mathbb{N}^V$, 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,

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

Given data $(x_i, y_i)_{i=1}^n$ we wish to infer the parameters $\theta, \alpha_1, \ldots, \alpha_K$.

1. **(2 points)** Write down the log likelihood for the model.
2. **(2 points)** Obtain the Maximum likelihood estimator for θ.
3. **(7 points)** Unfortunately, we cannot optimize $\alpha_1, \ldots, \alpha_K$ in closed form. We will do this via Newton's method. Obtain the Newton's method updates for α_1.
   In a realistic setting V is at least on the order of 10^4 and possibly as large as 10^7. Creating the $V \times V$ matrix let alone inverting it would be a terrible idea. Can you identify structure in your Hessian that allows you to perform the Newton's step efficiently? You should explain how the Newton step update can be performed using $\tilde{O}(V)$ time and memory. Here $\tilde{O}$ suppresses the dependence on n.
4. **(1 point)** Given a new point $x_* \in \mathbb{N}^d$, write the prediction rule.
5. **(3 points)** Now we adopt the Bayesian paradigm and assume that θ_0 was sampled from a $\text{Dir}(\theta_0)$ prior and all α_k were sampled from a $\mathcal{N}(0, \frac{1}{2\lambda} I_d)$ prior. Here $\theta_0 \in \mathbb{R}^K, \lambda \in \mathbb{R}_+$ are known. We wish to obtain the MAP estimate of the parameters. Derive the MAP estimate for θ and the Newton's method update for α_1. Show that the Newton update can still be performed efficiently.

   Using a normal prior for α_k is technically incorrect since it needs to be non-negative. But the math still works out even if you use a normal prior. It gives a quadratic penalty term which acts as a regularizer. Thanks to James Duyck for pointing this out.

2.2 Experiment (10 points)

Here we will do a simple experiment for the above problem. We recommend that you use Matlab for this part of the homework. We have provided you some starter code to load the data and output the results. Also, Matlab has vectorized implementations of the poly-gamma and log-gamma functions so it will make your life easier.

You need to implement code to

1. Estimate θ and $\alpha_k, k = 1, \ldots, K$. In particular, α_k should be estimated using Newton's method in $\tilde{O}(V)$ time.
2. Compute the prediction for a new point for a given model.

Please submit your code with the homework.

You have 2 datasets to experiment with given in `data1.mat` and `data2.mat`. The first is a simple dataset with $V = 4, n = 200, K = 3$ and the latter is slightly more challenging with $V = 1000, n = 100000, K = 3$. For both cases, you have been given the $n \times V$ training matrix `XTrain` and the $n \times 1$ training labels `yTrain`. In addition there is also test data and labels in `XTest` and `yTest`. 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 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 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 Newton's method should be sufficient.
- If you are using Matlab the following functions would be useful. `psi`, `gammaln`, `bsxfun`.

The following was the output of our implementation for `data1.mat`. We haven't thoroughly sanity checked our code so if your answers do not correspond first double check with your peers and then speak to us.

```
>> q2

theta =
    0.3596
    0.3399
    0.3005

alpha =
    5.8842    1.3622    1.1133    1.1900
    1.0761    4.3095    0.9938    1.0325
    1.0451    0.8546    2.3781    3.8601

preds =
     2
     1
     2
     2
     3

Accuracy: 1.0000
>>
```

3 Duality (Veeru)

3.1 Weak Duality

Consider the following problem with $f, h_1, h_2 : \mathbb{R}^d \to \mathbb{R}$:

$$\begin{aligned} &\inf_x f(x) \\ \text{s.t } &h_1(x) \leq 0, \\ &h_2(x) = 0 \end{aligned}$$

1. **(2 points)** Write the Lagrange dual $L(x, \lambda, u)$ where $\lambda \geq 0, u \in \mathbb{R}$ are the dual variables for the inequality and the equality respectively.
2. **(2 points)** Write down the dual function $g(\lambda, u)$ symbolically.
3. **(6 points)** Show that
$$\inf_{x \in P} f(x) \geq \sup_{\lambda \geq 0, u} g(\lambda, u).$$
In other words, the optimal primal value is $\geq$ the optimal dual value. The result holds when we have multiple inequality and equality constraints. This is called weak duality.

3.2 Optimal Coding

We will solve a simple convex problem in this part. The alphabet of a language consists of letters α_i, $i \in [n]$ where $[n] = \{1, 2, \cdots, n\}$. Assume that the letters occur with probabilites p_i in natural usage. We would like to find an optimal encoding of the letters in terms of bits, so that the expected number of bits used per letter is minimized, while still being able to decode the sequence of bits sent in a stream without any

markers between letters. For example, if there are $n = 32$ letters in the alphabet and all letters occur with equal probability($p_i = 1/32$), it intuitively makes sense to say that the optimal coding requires 5 bits for each letter. It turns out that optimal coding for the alphabet can be found by solving (1) given below. The objective function is the expected number of bits per letter. The first inequality in the problem is called Kraft's inequality and it ensures decodability of the letters.

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} \sum_{i=1}^{n} p_i x_i \\ &\text{s.t} \sum_{i=1}^{n} 2^{-x_i} \leq 1 \\ &\quad x_i \geq 0 \quad \forall i \in [n] \end{aligned} \tag{1}$$

Note that even though x_i should be positive integers, we are optimizing over all positive reals.

1. **(4 points)** Show that the feasible region is convex in (1).
2. **(4 points)** Show that equality holds in the first constraint for the optimal solution to (1).
3. **(7 points)** By the previous part, we can solve (1) with equality in the first constraint instead of inequality. Solve this new problem. Hint: You may show that the functions in the constraints are convex to justify that the gradient of Lagrangian is its subgradient.

Note:If you have already written up your solution with n in the base of the Kraft's inequality, you do not have to change it.

4 SVM and Perceptron (Veeru)

4.1 Finding support vectors from dual variables (6 points)

Consider the SVM dual problem:

$$\begin{aligned} &\min_{\alpha \in \mathbb{R}^n} \frac{1}{2} \sum_{i,j \in [n]} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle - \sum_{i=1}^{n} \alpha_i \\ &\text{s.t} \sum_{i=1}^{n} \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C \ \forall i \in [n] \end{aligned}$$

where α_i are the dual variables and $(x_i, y_i) \in \mathbb{R}^d \times \{-1, 1\}$ are feature, label pairs. Let $f(x) = \langle w, x \rangle + b$ be the prediction function, where $w \in \mathbb{R}^d, b \in R$ are primal variables. Argue from KKT conditions why the following hold:

$$\begin{aligned} \alpha_i = 0 &\Rightarrow y_i f(x_i) \geq 1 \\ 0 < \alpha_i < C &\Rightarrow y_i f(x_i) = 1 \\ \alpha_i = C &\Rightarrow y_i f(x_i) \leq 1 \end{aligned}$$

4.2 Using libsvm (8 points)

Use libsvm(`http://www.csie.ntu.edu.tw/ cjlin/libsvm/`) on a simple 2D dataset and report your observations as you increase the regularization parameter C from a small value to a large value. You may generate the dataset by sampling 50 points each from $\mathcal{N}(-\mu, \sigma^2 I)$ and $\mathcal{N}(\mu, \sigma^2 I)$ with $\mu = (1, 0)$ and $\sigma^2 = 0.8$(You may experiment with the value of σ^2 to get nice plots). Label the points drawn from one distribution -1 and those from the other $+1$. Plot how the learned plane and margin in your answer change with C. Also, plot

testing error vs. training error with varying C, holding out a randomly chosen half of the data for testing. You may experiment with about 8 values of C for nice plots.

4.3 Mistake bound for Perceptron

Suppose we run the perceptron algorithm on a dataset $\{(x_i, y_i) \in \mathbb{R}^d \times \{-1, 1\} | i \in [n]\}$ which is separable by a margin, that is, $\exists w_*, \delta$ such that $y_i \langle x_i, w_* \rangle \geq \delta > 0, \forall i \in [n]$ and $\|w_*\| = 1$. Assume that $\|x_i\| \leq M, \forall i \in [n]$. Let $w_0 = 0, w_1, \cdots, w_k, \cdots$ be the iterates of the algorithm. Let the update step be $w_{k+1} = w_k + y_{i(k+1)} x_{i(k+1)}$ where the perceptron fails on $i(k+1)$th data point with w_k. The perceptron with model w, is assumed to fail on a data point (x, y), if $\langle w, yx \rangle \leq 0$. The bias term is incorporated into all x_i by setting their first coordinate to 1.

1. **(3 points)** Show that $\langle w_k, w_* \rangle \geq k\delta$.
2. **(4 points)** Show that $\|w_k\|^2 \leq kM^2$.
3. **(4 points)** Show that the number of mistakes done by the algorithm $k_* \leq \frac{M^2}{\delta^2}$.