1072 Deep Learning – Homework 1

Due: April 12, 2019, 11:55pm

1. (10%) (Maximum Likelihood Estimation) Please set the derivatives of the log likelihood function of a Gaussian Probability Distribution Function $\log p(x;u,\sigma^2) = -\frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i-u)^2 - \frac{N}{2}log\sigma^2 - \frac{N}{2}\log(2\pi)$ to zero with respect to u and σ and verify the following results below:

$$u_{ML} = \underset{u}{\operatorname{argmin}} - \sum_{i=1}^{N} \log p(x;u,\sigma^2) = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$\sigma_{ML} = \underset{\sigma}{\operatorname{argmin}} - \sum_{i=1}^{N} \log p(x;u,\sigma^2) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-u_{ML})^2}$$

Ans:

Let $L(x;u,\sigma^2) = \log p(x;u,\sigma^2)$

Set the derivative of $L(x;u,\sigma^2)$ with respect to u to 0

$$\frac{\partial L(x;u,\sigma^2)}{\partial u} = \frac{\partial - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i-u)^2}{\partial u} = \frac{1}{\sigma^2}\sum_{i=1}^{N}(x_i-u) = 0$$

$$u_{ML} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Set the derivative of $L(x;u,\sigma^2)$ with respect to σ to 0

$$\frac{\partial L(x;u,\sigma^2)}{\partial \sigma} = \frac{\partial - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i-u)^2 - \frac{N}{2}log\sigma^2}{\partial \sigma} = \frac{1}{\sigma^3}\sum_{i=1}^{N}(x_i-u)^2 - N\frac{1}{\sigma} = 0$$

$$\sigma_{ML} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-u_{ML})^2}$$

2. (10%) Please load 'data.mat' into your Matlab or Python code, where you will find $x, y \in R^{1001}$. Now do the following procedures, paste your source code and show the results in your report.
   2.1. Plot the data using plot function.

   >> plot(x, y); grid

   2.2. Compute the least square line $y = \theta_0 + x\theta_1$ using the given data and overlay the line over the given data.
   >> hold on; plot(x, θ_0+θ_1*x, '--')

Ans:

```
load data.mat;
plot(x, y); grid
A = [x.^0 x.^1];
sol = inv(A'*A)*A'*y;
hold on; plot(x, sol(1)+sol(2)*x, --);
title(Least squares line fitting); xlabel(x); ylabel(y);
```

3. (10%) Using the same data from Question 2, compute the least square parabola (i.e. second order polynomial $y = \theta_0 + x\theta_1 + x^2\theta_2$) to fit the data. (5%) Explain which formulation (line or parabola) is more suitable for this dataset and why? (paste your source code and show the results in your report)

Ans:

```
load data.mat;
figure; plot(x,y,'g');
A=[x.^0 x.^1 x.^2];
sol = inv(A'*A)*A'*y;
hold on; plot(x, A*sol, --);
xlabel('x'); ylabel('y');
```

4. (20%) Using the same data from Question 2, now we use the loss function (L1 Norm) below instead of least square based methods. (paste your source code and show the results in your report) Hint: use a gradient descent approach.

$$Loss\ Function: f(\boldsymbol{\theta}) = \sum_{i=1}^{N} |y_i - (\theta_0 + x\theta_1 + x^2\theta_2)|$$

Ans:

```
load data.mat;
A = [x.^0 x.^1 x.^2];
alpha = 0.0001;
```

```
n_iter = 10000;
coeff = rand(3,1);
for i = 1:n_iter
    signs = sign(y-A*coeff);
    grad = signs'*(-A);
    coeff = coeff-alpha*grad';
end
```

5. (25%) In ‘train.mat,’ you can find 2-D points X=[x1, x2] and their corresponding labels Y=y. Please use logistic regression $h(\boldsymbol{\theta}) = \frac{1}{1+e^{-\boldsymbol{\theta}^T x}}$ to find the decision boundary (optimal $\boldsymbol{\theta}^*$) based on ‘train.mat.” Report the test error on the test dataset ‘test.mat.’ (percentage of misclassified test samples) Hint: you can use “mnrfit” in Matlab or “LogisticRegression” in Python.

Ans:

```
load('train.mat');
X = [x1,x2];
Y = y;
scatter(x1(Y==1),x2(Y==1),'r','+');
hold on;
scatter(x1(Y==0),x2(Y==0),'b','+');
B = mnrfit(X, Y+1); % For “mnrfit”, the labels should start from 1, so we add 1 to the labels
x = [min(x1)-0.1:0.01:max(x1)+0.1];
y = (-B(1)-B(2)*x)./B(3);
hold on;
plot(x,y,'k');
axis([min(x1)-0.1,max(x1)+0.1,min(x2)-0.1,max(x2)+0.1]);

%%%%%%%%%%%%%%%%%%%%%%%%

% test your model on test dataset

load('test.mat');
X = [x1,x2];
Y = y;
pihat = mnrval(B,X);
[˜, index] = max(pihat');
predict_label = index-1;
test_error = sum(predict_label'˜=Y)/length(Y);
```

6. (20%) Please use a gradient descent method to solve Question 5. (show your code, decision boundary, and test error on the test dataset)

Ans:

```
load('train.mat');
X = [x1, x2];
Y = y;
```

```
scatter(x1(Y==1), x2(Y==1), 'r', '+');
hold on;
scatter(x1(Y==0), x2(Y==0), 'b', '+');

N = size(y, 1);
A = cat(2, ones(N, 1), X); %[1 x1 x2]
w = zeros(3, 1);
alpha = 0.01;
epsilon = 1e-3;
maxiter = 1e3;
fx = zeros(maxiter, 1);
for iter=1:maxiter
    fx(iter) = Y'*log(1+exp(-A*w))+(1-Y)'*log(1+exp(A*w));
    grad = A'*(1./(1+exp(-A*w))-Y);
    if norm(grad) < epsilon
        break;
    end
    w = w -alpha*grad;
end
```