CS 8725: Report for assignment 3

Chanmann Lim

September 28, 2015

The Matlab code for all experiments is in the **Appendix** section.

Programming 1: We are given a bunch of data samples of "Average height and weight of American women aged 30 to 39" and the task is to design linear regression algorithm to predict the *weight* $\in \mathbb{R}$ denoted by Y from a given height measurement denoted by X then the problem becomes $f(X) \to Y$ finding a function mapping from X to Y. In linear regression we assume that $f(X)$ take a linear form with respect to X that is $f(X) = \beta_1 + \beta_2 X$ and the goal is to choose $\hat{f}(X)$ that minimizes the prediction error (squared error).

$$\hat{f}_n^L = \underset{f}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} (f(X_i) - Y_i)^2 \tag{1}$$

By rewriting $f(X)$ in term of $\beta = [\beta_1 \ \beta_2]^T$, minimizing $f(X)$ becoming minimizing β.

$$\begin{align} \hat{\beta} &= \underset{\beta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} (X_i \beta - Y_i)^2 \tag{2} \\ &= \underset{\beta}{\operatorname{argmin}} \frac{1}{n} (A\beta - Y)^T (A\beta - Y) \tag{3} \end{align}$$

Where,

$$A = \begin{bmatrix} 1 & X_1 \\ \vdots & \vdots \\ 1 & X_n \end{bmatrix} \text{ and } Y = \begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix} \tag{4}$$

If we define $J(\beta) = (A\beta - Y)^T (A\beta - Y)$ then minimizing $J(\beta)$ is equivalent to (3).

$$\begin{align} \frac{\partial J(\beta)}{\partial \beta} &= \frac{\partial A^T A \beta \beta^T - 2\beta^T A^T Y - Y^T Y}{\partial \beta} \tag{5} \\ &= 2A^T A \beta - 2A^T Y \tag{6} \end{align}$$

$$\begin{align} 2A^T A\hat{\beta} - 2A^T Y = 0 \tag{7} \\ A^T A \hat{\beta} = A^T Y \tag{8} \\ \hat{\beta} = (A^T A)^{-1} A^T Y \tag{9} \end{align}$$

Eq. (9) is the normal equation with $(A^T A)$ as normal matrix and $\hat{f}_n^L = X\hat{\beta}$. As a result we obtained the linear regression coefficient in (10) and the expected error in (13):

$$\hat{\beta} = [-39.0620 \quad 61.2722]^T \tag{10}$$

$$R(\hat{f}) = \mathbb{E}[(\hat{f}(X) - Y)^2] \tag{11}$$
$$= \frac{1}{n}(X\hat{\beta} - Y)^T(X\hat{\beta} - Y) \tag{12}$$
$$= 0.4994 \tag{13}$$

In fact for this case in particular, we know that we could further reduce the expected error $R(\hat{f})$ by introducing quadratic term into the regression. With this procedure we have turned linear regression into polynomial regression and by following the derivation in solving for $\hat{\beta}$ it turn out that $\hat{\beta} = (A^TA)^{-1}A^TY$ where $\beta = [\beta_1\ \beta_1\ \beta_3]^T$ and

$$A = \begin{bmatrix} 1 & X_1 & X_1^2 \\ \vdots & \vdots & \vdots \\ 1 & X_n & X_n^2 \end{bmatrix}$$

We obtained $\hat{\beta} = [128.8128 \quad -143.1620 \quad 61.9603]^T$ and $R(\hat{f}) = 0.0506$ for second order polynomial regression. Figure 1 show the both linear and polynomial regression functions with respect to the data samples.

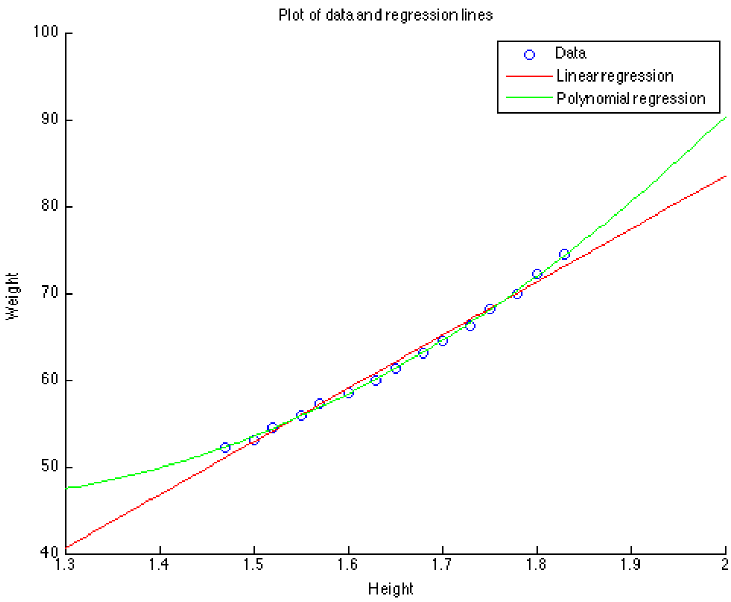

Figure 1: Plot of data and regression lines

Appendix:

assignment_3.m

```
clc;
clear all;
close all;

problem_1
problem_2
```

problem_1.m

```
% Load data
x = [1.47 1.5 1.52 1.55 1.57 1.6 1.63 1.65 1.68 1.7 1.73 1.75 1.78 1.8 1.83]';
y = [52.21 53.12 54.48 55.84 57.2 58.57 59.93 61.29 63.11 64.47 66.28 68.1 69.92 72.19 74.46]';

% Linear regression
A = [ones(length(x), 1) x];
Beta = (A'*A)\A'*y;
display(Beta);

% Evaluation
prediction = A*Beta;
expected_loss_beta = (y-prediction)' * (y-prediction) / length(y);
display(expected_loss_beta);

% Polynomial regression
A = [A x.^2];
Beta_2 = (A'*A)\A'*y;
display(Beta_2);

% Evaluation
prediction = A*Beta_2;
expected_loss_beta2 = (y-prediction)' * (y-prediction) / length(y);
display(expected_loss_beta2);

% Plot
figure;
scatter(x, y); hold on;
x_value = linspace(1.3, 2);
plot(x_value, Beta(1) + Beta(2)*x_value, 'r');
plot(x_value, Beta_2(1) + Beta_2(2)*x_value + Beta_2(3)*x_value.^2, 'g');
hold off;
title('Plot of data and regression lines');
xlabel('Height');
ylabel('Weight');
legend('Data', 'Linear regression', 'Polynomial regression');
```

problem_2.m

```
% Load data
[x1,x2,x3,x4] = textread('iris.data', '%f,%f,%f,%f,%*s');
X = [ones(length(x1),1) x1 x2 x3 x4];

setosa = 1:50;
versicolor = 51:100;
[~, d] = size(X);
y = [zeros(length(setosa),1); ones(length(versicolor),1)];

% Logistic regression gradient ascent
% initialize weight = [-1, 1]
W = 2*rand(d, 1)-1;
display(W);
% maximum iteration
T = 100000;
% learning rate
alpha = 0.001;
% conditional log likelihood
lw = zeros(1, T);

for t=1:T
    prediction = exp(X*W) ./ (1+exp(X*W));
```

```
    W = W + alpha * X' * (y - prediction);
    lw(t) = y'*X*W - sum( log(1+exp(X*W)) );
    if t > 1
        delta_lw = abs( (lw(t) - lw(t-1)) / lw(t-1) );
        % convergence criterion = percentage change < 0.03%
        if delta_lw < 3e-4
            break;
        end
    end
end

% Evaluation
classification = X*W > 0;
accuracy = sum(classification == y) / length(y);
display(accuracy);

% Plot of conditional log likelihood
figure;
plot(1:t, lw(1:t));
title(['Conditional log likelihood score (finish at t=' num2str(t) ')']);
xlabel('Iterations');
ylabel('l(w)');
```