1 Newton's method for computing least squares

(a) Hessian (H) is a matrix of second derivatives. We can start from the expression for a derivative of cost function $J(\Theta)$ as was presented in the lecture notes:

$$\frac{\partial}{\partial \Theta_j} J(\Theta) = (h_\Theta(x) - y)x_j \rightarrow \nabla_\Theta J(\Theta) = X^T X \Theta - X^T \vec{y}$$

where $h_\Theta(x) = \sum_{i=0}^{n} \Theta_i x_i$, and differentiate it second time wrt Θ_k:

$$\frac{\partial^2}{\partial \Theta_j \partial \Theta_k} J(\Theta) = x_j x_k$$

This can be also expressed in a matrix form: $H = X^T X$.

(b) The step in the Newton's method is defined:

$$\Theta := \Theta - \frac{f(\Theta)}{f'(\Theta)}$$

when one looks for roots of $f(\Theta)$. Since we want to minimize $J(\Theta)$, we want $\nabla_\Theta J(\Theta) = 0$, and thus we use the modified version expressed on matrices and vectors:

$$\Theta := \Theta - H^{-1} \nabla_\Theta J(\Theta)$$

. Let us now look how looks Θ after first step (Θ_1) taking some Θ_0 as a starting point, and implementing the expressions found in the previous exercise:

$$\Theta_1 = \Theta_0 - (X^T X)^{-1}(X^T X \Theta_0 - X^T \vec{y}) = \Theta_0 - \Theta_0 + (X^T X)^{-1} X^T \vec{y} = (X^T X)^{-1} X^T \vec{y}$$

As we can see our expression do not depend on the choice of Θ_0, and we arrive to the solution in one step.

2 Locally-weighted logistic regression

(a) The program written for this exercise is available in q2/ directory, here we present the function lwlr.m:

```
function y = lwlr(X_train, y_train, x, tau)

%% Locally weighted logistic regression
% the vaule of parameter lambda given in exerscise
lam = 0.0001;
%% m the sample size
m = length(y_train);
```

```
%% Update of Theta in Newton's method Th := Th + l'(Th)/l''(Th)
%% Initialise Th as a zero vector:
Th = [0;0];
Grad = [1;1];

%% to avoid too many iterations
iter = 0;
%% building vector of weights for the training set
weights = exp(-sum((X_train - repmat(x', m, 1)).^2, 2) / (2 * tau^2));

while (norm(Grad) > 0.0001 && iter < 10000)
  iter = iter+1;
%% vector of h_Theta from X
  h = 1.0 ./ (1.0 + exp(-sum(X_train .* Th',2)));

%%  auxiliary variables z and D for gradient and hessian calculation
  z = weights .* (y_train - h);
  D = diag(- weights .* (1.0 - h) .* h);

  %% Hessian expression H = XT DX  I
  Hess = X_train' * D * X_train - lam * eye(2);
  %% Grad expression l() = XT z
  Grad = X_train' * z - lam * Th;
  %% Update of Theta in Newton's method Th := Th + l'(Th)/l''(Th)
  Th = Th - Hess \ Grad;
endwhile
%% make prediction for y
if (1.0 / (1.0 + exp(-x' * Th)) > 0.5)
  y = 1;
else
  y = 0;
endif
```

(b) There will be pics at some point...

3 Multivariate least squares

(a) We have a cost function:

$$J(\Theta) = \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{p} ((\Theta^T x^{(i)})_j - y_j^{(i)})$$

. To express this in a vector/matrix form let us consider building blocks: the design matrix $X \in \mathbb{R}^{m \times n}$, the target matrix $Y \in \mathbb{R}^{m \times p}$ and the parameters matrix $\Theta \in \mathbb{R}^{n \times p}$. The cost function expressed with these matrices:

$$J(\Theta) = \frac{1}{2}\mathbf{tr}(X\Theta - Y)^T(X\Theta - Y)$$

where matrices have right dimensions that the operations are well defined and the trace is applied to take the sum of diagonal elements ($\sum_j$ in first expression).

(b) The minimization of $J(\Theta)$ requires $\nabla_\Theta J(\Theta) = 0$:

$$\nabla_\Theta J(\Theta) = \frac{1}{2}\nabla_\Theta \mathbf{tr}(X\Theta - Y)^T(X\Theta - Y) = \frac{1}{2}\nabla_\Theta \mathbf{tr}(\Theta^T X^T X\Theta - Y^T X\Theta - \Theta^T X^T Y + Y^T Y) =$$

now we use properties of trace (eg. cyclic or $\mathbf{tr}A = \mathbf{tr}A^T$) and of the matrix derivatives as shown in the Sec. 2.1. of lecture notes:

$$= \frac{1}{2}\nabla_\Theta \mathbf{tr}\Theta^T X^T X\Theta - \nabla_\Theta \mathbf{tr}Y^T X\Theta = X^T X\Theta - X^T Y$$

We identify it with zero and solve for Θ:

$$X^T X\Theta - X^T Y = 0$$

$$X^T X\Theta = X^T Y$$

$$\Theta = (X^T X)^{-1} X^T Y$$

(c) When there is just one value of y, we get the expression (from the lecture notes):

$$\theta = (X^T X) X^T \vec{y}$$

. When we consider the p independent target vectors $\vec{y_j}$ where $j = 1, ..., p$ we can combine them in a matrix $Y = (\vec{y_1}...\vec{y_p})$ and also the paramters $\theta \to \theta_j$ will arrive into the final parameter matrix $\Theta = [\theta_1...\theta_p]$.

4 Naive Bayes

(a)

5 Exponential family and the geometric distribution

(a) Geometric distribution given by:

$$p(y; \phi) = (1 - \phi)^{(y-1)}\phi \quad \text{for} \quad y = 1, 2, 3, ...$$

and the Exponential Family form:

$$p(y;\eta) = b(y)\exp(\eta^T T(y) - a(\eta))$$

where $T(y)$ is a sufficient statistic, usually $T(y) = y$, and η is natural parameter. Let us play with the form of the geometric distribution to allow the identification of the Exponential Family form:

$$p(y;\phi) = \exp(\log((1-\phi)^{(y-1)}\phi) = \exp((y-1)\log(1-\phi)+\log(\phi)) = \exp(y\log(1-\phi)-\log(\frac{\phi}{1-\phi}))$$

We can now identify:

$$T(y) = y$$

$$b(y) = 1$$

$$\eta^T = \log(1-\phi)$$

from this we can get: $\phi = 1 - \exp(\eta)$,

$$a(\eta) = -\log(\frac{\phi}{1-\phi}) = -log(\frac{1-\exp(\eta)}{1-1+\exp(\eta)}) = \log(\frac{1}{1+\exp(-\eta)})$$

(b) knowing $E[y;\phi] = \frac{1}{\phi}$, we have:

$$g(\eta) = E[T(y);\eta] = E[y;\eta(\phi)] = \frac{1}{phi} = \frac{1}{1-\exp(\eta)}$$

(c) having $p(y;\phi) = (1-\phi)^{(y-1)}\phi$ and $\phi = 1 - \exp(\Theta^T x)$ we get the log-likelihood of a form:

$$l(\Theta) = \log(p(y^{(i)}|x^{(i)};\Theta) = \log(1-\phi)^{(y-1)}\phi) = \log((\exp(\Theta^T x^{(i)})^{y^{(i)}-1}(1-\exp(\Theta^T x)) = (y^{(i)}-1)\Theta^T x^{(i)} + \log(1-e$$

$$= y^{(i)}\Theta^T x^{(i)} + \log(\frac{1-\exp(\Theta^T x^{(i)})}{(\exp(\Theta^T x^{(i)})} = y^{(i)}\Theta^T x^{(i)} + \log(\exp(-\Theta^T x^{(i)}) - 1)$$

To derive rule for a stochastic gradient ascent, we need to compute the gradient:

$$\frac{d}{d\Theta_j} l(\Theta) = y^{(i)}x_j^{(i)} + \frac{\exp(-\Theta^T x^{(i)})(-x_j^{(i)})}{\exp(-\Theta^T x^{(i)}) - 1} = (y^{(i)} - \frac{1}{1-\exp(\Theta^T x^{(i)})})x_j^{(i)}$$

and then the rule is:

$$\Theta_j := \Theta_j + \alpha\frac{\partial l(\Theta)}{\partial\Theta_j}$$

$$\Theta_j := \Theta_j + \alpha(y^{(i)} - \frac{1}{1-\exp(\Theta^T x^{(i)})})x_j^{(i)}$$