# 6.867 Machine Learning

### Solutions for Problem Set 2

Wednesday, October 8

## Problem 1: active regression

**(1-1)** (5pts) Recall that $\hat{w} \sim N(w^*, (X'X)^{-1})$ where $w^*$ are the true parameter values (under the assumed model of how the outputs $y$ were randomly generated given the inputs $x$). Also, it was shown how this implies that $E\{f(x,\hat{w})|x\} = f(x;w^*)$ where $f(x;w) = w'\phi(x)$. Now, we compute the variance of $f(x;\hat{w})$ given $x$.

$$\begin{aligned}
\mathrm{var}(f(x,\hat{w})|x) &= E\{(f(x,\hat{w}) - f(x,w^*))^2|x\} && (1)\\
&= E\{(\phi'(x)(\hat{w}-w^*))^2|x\} && (2)\\
&= E\{\phi'(x)(\hat{w}-w^*)(\hat{w}-w^*)'\phi(x))|x\} && (3)\\
&= \phi'(x)E\{(\hat{w}-w^*)(\hat{w}-w^*)'\}\phi(x) && (4)\\
&= \phi'(x)(X'X)^{-1}\phi(x) && (5)
\end{aligned}$$

**(1-2)**

(a) (3pts) Recall that $X = (\phi(x_1)\ldots\phi(x_n))'$ where $\phi(x)$ is an $m \times 1$ column vector. Then,

$$\begin{aligned}
X'X &= (\phi(x_1)\ldots\phi(x_n))\begin{pmatrix}\phi'(x_1)\\ \vdots \\ \phi'(x_n)\end{pmatrix} && (6)\\
&= \sum_{i=1}^{n}\phi(x_i)\phi'(x_i) && (7)
\end{aligned}$$

This is clearly symmetric, $(X'X)' = X'X'' = X'X$. Also, for $v \in R^m$

$$\begin{aligned}
v'(X'X)v &= \sum_{i=1}^{n} v'\phi(x_i)\phi'(x_i)v && (8)\\
&= \sum_{i}(v'\phi(x_i))^2 && (9)\\
&\geq 0 && (10)
\end{aligned}$$

so that $(X'X)$ is positive semi-definite.

(b) (2pts) Let $\phi(x) = (1, x)'$, then

$$X'X = \sum_i \begin{pmatrix} 1 \\ x_i \end{pmatrix} (1 \;\; x_i) \tag{11}$$

$$= \sum_i \begin{pmatrix} 1 & x_i \\ x_i & x_i^2 \end{pmatrix} \tag{12}$$

$$= \begin{pmatrix} n & \sum_i x_i \\ \sum_i x_i & \sum_i x_i^2 \end{pmatrix} \tag{13}$$

**(1-3)** (5pts) Let $x$ be distributed according to a symmetric distribution $Q(x)$. By symmetry, we must have $E_Q\{\mathrm{x}\} = 0$. The variance is $E_Q\{x^2\} = v^2$. Then,

$$E_Q\{\phi(x)\phi'(x)\} = E_Q\left\{\begin{pmatrix} 1 & x \\ x & x^2 \end{pmatrix}\right\} \tag{14}$$

$$= \begin{pmatrix} 1 & 0 \\ 0 & v^2 \end{pmatrix} \tag{15}$$

Evaluate the objective function $J(X)$ from PS2 Eq. (10). First, rewrite $X'X$ as

$$X'X = n \begin{pmatrix} 1 & \langle\, x \,\rangle \\ \langle\, x \,\rangle & \langle\, x^2 \,\rangle \end{pmatrix} \tag{16}$$

where $\langle\, \cdot \,\rangle$ denotes the average over the query points $\langle\, x \,\rangle = \frac{1}{n}\sum_i x_i$ and $\langle\, x^2 \,\rangle = \frac{1}{n}\sum_i x_i^2$. Now,

$$J(X) = \frac{1}{n}\mathrm{trace}\left\{\begin{pmatrix} 1 & \langle\, x \,\rangle \\ \langle\, x \,\rangle & \langle\, x \,\rangle^2 \end{pmatrix}^{-1} \begin{pmatrix} 1 & 0 \\ 0 & v^2 \end{pmatrix}\right\} \tag{17}$$

$$= \frac{1}{n}\mathrm{trace}\left\{\frac{1}{\langle\, x^2 \,\rangle - \langle\, x \,\rangle^2} \begin{pmatrix} \langle\, x \,\rangle^2 & -\langle\, x \,\rangle \\ -\langle\, x \,\rangle & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & v^2 \end{pmatrix}\right\} \tag{18}$$

$$= \frac{1}{n}\mathrm{trace}\left\{\frac{1}{\langle\, x^2 \,\rangle - \langle\, x \,\rangle^2} \begin{pmatrix} \langle\, x^2 \,\rangle & -v^2\langle\, x \,\rangle \\ -\langle\, x \,\rangle & v^2 \end{pmatrix}\right\} \tag{19}$$

$$= \frac{1}{n}\left(\frac{\langle\, x^2 \,\rangle + v^2}{\langle\, x^2 \,\rangle - \langle\, x \,\rangle^2}\right) \tag{20}$$

Note that the denominator is just the sample variance $\langle\, (x - \langle\, x \,\rangle)^2 \,\rangle = \langle\, x^2 \,\rangle - \langle\, x \,\rangle^2$.

**(1-4)** (10pts) Subject to the constraint $x \in [-1, +1]$, how should we choose $(x_1, \ldots, x_n)$ so as to make $J(X)$ as small as possible? Let's write $n \cdot J(X)$ in terms of the sample mean and sample variance:

$$n \cdot J(X) = \frac{[\langle\, (x - \langle\, x \,\rangle)^2 \,\rangle + \langle\, x \,\rangle^2] + v^2}{\langle\, (x - \langle\, x \,\rangle)^2 \,\rangle} = 1 + \frac{\langle\, x \,\rangle^2 + v^2}{\langle\, (x - \langle\, x \,\rangle)^2 \,\rangle} \tag{21}$$

where we have used the fact that $\langle\, x^2 \,\rangle = \langle\, (x - \langle\, x \,\rangle)^2 \,\rangle + \langle\, x \,\rangle^2$. Apparently, we should like to minimize the squared sample mean $\langle\, x \,\rangle^2$ while simultaneously maximizing the sample

variance $\langle\,(x - \langle\,x\,\rangle)^2\,\rangle$ (if this is possible, then it is optimal). For $n$ even, the sample variance is maximized when half the inputs are at $x = -1$ and half are at $x = +1$. Then, the mean is zero and the sample variance is one. But this also minimizes the squared sample mean. Hence, this achieves the smallest possible value of $J(X)$ consistent with the constraints, e.g. $J(X) = (1 + v^2)/n$. This proves that the proposed strategy is optimal. Any deviation from this strategy will either decrease the variance (while keeping the mean fixed at zero) or both decrease the variance and increase the squared sample mean. Hence, any other strategy is suboptimal.

It would have been fine to assume here that by symmetry the points we choose have to have zero mean, i.e., $\langle\,x\,\rangle = 0$.

**(1-5)** (optional) If $Q(x)$ no longer has to be symmetric around zero, then we can simply put a point mass on $x = 1$. In other words, any sample $x \sim Q(x)$ would be 1 with probability one. So we are only interested in the value of the function at $x = 1$ in this case. Querying any other point would be useless.

## Problem 2: active parameter estimation

**(2-1)** (10pts) We wish to choose $\theta = (\theta_{1|0}, \theta_{1|1})$ to minimize the (conditional) log-likelihood:

$$\begin{aligned} l(\theta) &= \log P(y^{(n)}|x^{(n)}, \theta) && (22)\\ &= \sum_{i|x_i=0} \log P(y_i|x=0, \theta_{1|0}) + \sum_{i|x_i=1} \log P(y_i|x=1, \theta_{1|1}) && (23)\\ &= l_0(\theta_{1|0}) + l_1(\theta_{1|1}) && (24) \end{aligned}$$

This decomposes into two separate optimization problems: choose $\theta_{1|0}$ to miniminize $l_0$; choose $\theta_{1|1}$ to minimize $l_1$.

To optimize $l_0$ w.r.t. $\theta_{1|0}$, first rewrite $l_0$ as

$$l_0(\theta_{1|0}) = N_{\mathrm{x,y}}(0,1) \log \theta_{1|0} + (N_{\mathrm{x}}(0) - N_{\mathrm{x,y}}(0,1)) \log(1 - \theta_{1|0}) \tag{25}$$

Then, compute derivative of $l_0$ w.r.t. $\theta_{1|0}$ and set to zero.

$$\frac{\partial l_0}{\partial \theta_{1|0}} = \frac{N_{\mathrm{x,y}}(0,1)}{\theta_{1|0}} - \frac{N_{\mathrm{x}}(0) - N_{\mathrm{x,y}}(0,1)}{1 - \theta_{1|0}} = 0 \tag{26}$$

Solving for $\theta_{1|0}$ gives ML-estimate.

$$\hat{\theta}_{1|0} = \frac{N_{\mathrm{x,y}}(0,1)}{N_{\mathrm{x}}(0)} \tag{27}$$

By a similar calculation:

$$\hat{\theta}_{1|1} = \frac{N_{\mathrm{x,y}}(1,1)}{N_{\mathrm{x}}(1)} \tag{28}$$

We show that $\hat{\theta}_{1|0}$ is an unbiased estimate of $\theta_{1|0}$. Take (conditional) expectation over all possible configurations $y^{(n)}$ with $x^{(n)}$ held fixed (note that $N_{\mathrm{x}}(0) = \sum_i (1 - x_i)$ is *not* random).

$$\begin{aligned} E\{\hat{\theta}_{1|0}|x^{(n)}\} &= E\left\{ \frac{N_{\mathrm{x,y}}(0,1)}{N_{\mathrm{x}}(0)} \Big| x^{(n)} \right\} && (29) \\ &= \frac{1}{N_{\mathrm{x}}(0)} E\left\{ \sum_{i|x_i=0} y_i \Big| x^{(n)} \right\} && (30) \\ &= \frac{1}{N_{\mathrm{x}}(0)} \sum_{i|x_i=0} E\{y_i|x_i = 0\} && (31) \\ &= \frac{1}{N_{\mathrm{x}}(0)} \sum_{i|x_i=0} \theta_{1|0} && (32) \\ &= \theta_{1|0} && (33) \end{aligned}$$

By a similar argument, $\hat{\theta}_{1|1}$ is an unbiased estimate of $\theta_{1|1}$.

Finally, show that $\hat{\theta}_{1|1}$ and $\hat{\theta}_{1|0}$ are uncorrelated. The covariance between two random variables $u$ and $v$ is $\mathrm{cov}(u,v) = E\{(u - E\{u\})(v - E\{v\})\} = E\{uv\} - E\{u\}E\{v\}$. This is zero ($u$ and $v$ are uncorrelated) if and only if $E\{uv\} = E\{u\}E\{v\}$. To show this for our two estimates, observe that $\hat{\theta}_{1|0}$ is only a function of those outputs $y_{|0} = (y_i|x_i = 0)$ where the input was $x = 0$; likewise, $\hat{\theta}_{1|1}$ only depends on those outputs $y_{|1} = (y_i|x_i = 1)$ for input $x = 1$. Then appeal to conditional independence of the outputs given the inputs, i.e.

$$P(y_{|0}, y_{|1}|x^{(n)}) = P(y_{|0}|x^{(n)})P(y_{|1}|x^{(n)}) \tag{34}$$

This implies

$$E\{\hat{\theta}_{1|0}(y_{|0})\hat{\theta}_{1|1}(y_{|1})|x^{(n)}\} = E\{\hat{\theta}_{1|0}(y_{|0})|x^{(n)}\}E\{\hat{\theta}_{1|1}(y_{|1})|x^{(n)}\} \tag{35}$$

hence the (conditional) covariance between the estimates (given $x^{(n)}$) is zero.

*Scoring:* 6pts for estimates, 2pts for unbiased, 2pts for uncorrelated.

**(2-2)** (10pts) Two important facts:

- The variance of a random variable $x$ scaled by a (non-random) constant $c$ is $\mathrm{var}(c\mathbf{x}) = c^2\mathrm{var}(x)$.
- The variance of a sum of independent random variables is just the sum of the variances, i.e. $\mathrm{var}(\sum_i \mathrm{x}_i) = \sum_i \mathrm{var}(\mathrm{x}_i)$.

We compute the variance of $\hat{\theta}_{1|0}$ (again holding $x^{(n)}$ fixed, expectation is over all possible $y^{(n)}$):

$$\mathrm{var}(\hat{\theta}_{1|0}|x^{(n)}) = \mathrm{var}\left( \frac{N_{\mathrm{x,y}}(0,1)}{N_{\mathrm{x}}(0)} \Big| x^{(n)} \right) \tag{36}$$

$$= \frac{1}{N_{\mathrm{x}}(0)^2}\mathrm{var}\left(\sum_{i|x_i=0}\mathrm{y}_i\left|x^{(n)}\right.\right) \tag{37}$$

$$= \frac{1}{N_{\mathrm{x}}(0)^2}\sum_{i|x_i=0}\mathrm{var}(\mathrm{y}|\mathrm{x}=0) \tag{38}$$

$$= \frac{\mathrm{var}(\mathrm{y}|\mathrm{x}=0)}{N_{\mathrm{x}}(0)} \tag{39}$$

$$= \frac{\theta_{1|0}(1-\theta_{1|0})}{N_{\mathrm{x}}(0)} \tag{40}$$

where we have used the hint in the last line.

By a similar calculation:

$$\mathrm{var}(\hat{\theta}_{1|1}|x^{(n)}) = \frac{\theta_{1|1}(1-\theta_{1|1})}{N_{\mathrm{x}}(1)} \tag{41}$$

**(2-3)** (10pts) Let $V(x_{n+1}) = J(x^{(n)};\hat{\theta}) - J(x^{(n+1)};\hat{\theta})$, we wish to select $x_{n+1} = \arg\max_{x\in\{0,1\}} V(x)$. Evaluate $V$ for $x_{n+1}=0$.

$$V(0) = \left(\frac{\hat{\theta}_{1|0}(1-\hat{\theta}_{1|0})}{N_{\mathrm{x}}(0)} + \frac{\hat{\theta}_{1|1}(1-\hat{\theta}_{1|1})}{N_{\mathrm{x}}(1)}\right) - \left(\frac{\hat{\theta}_{1|0}(1-\hat{\theta}_{1|0})}{N_{\mathrm{x}}(0)+1} + \frac{\hat{\theta}_{1|1}(1-\hat{\theta}_{1|1})}{N_{\mathrm{x}}(1)}\right) \tag{42}$$

$$= \hat{\theta}_{1|0}(1-\hat{\theta}_{1|0})\left(\frac{1}{N_{\mathrm{x}}(0)} - \frac{1}{N_{\mathrm{x}}(0)+1}\right) \tag{43}$$

$$= \frac{\hat{\theta}_{1|0}(1-\hat{\theta}_{1|0})}{N_{\mathrm{x}}(0)(N_{\mathrm{x}}(0)+1)} \tag{44}$$

Likewise, for $x_{n+1}=1$ we have

$$V(1) = \frac{\hat{\theta}_{1|1}(1-\hat{\theta}_{1|1})}{N_{\mathrm{x}}(1)(N_{\mathrm{x}}(1)+1)} \tag{45}$$

We pick $x_{n+1}=1$ if $V(0)<V(1)$ and pick $x_{n+1}=0$ if $V(0)>V(1)$. Equivalently, pick $x_{n+1}=1$ if and only if

$$\frac{N_{\mathrm{x}}(1)(N_{\mathrm{x}}(1)+1)}{N_{\mathrm{x}}(0)(N_{\mathrm{x}}(0)+1)} < \frac{\hat{\theta}_{1|1}(1-\hat{\theta}_{1|1})}{\hat{\theta}_{1|0}(1-\hat{\theta}_{1|0})} \tag{46}$$

(otherwise pick $x_{n+1}=0$). Note that the right hand side of this comparison converges to a constant (a characteristic of the underlying true conditional model $P(y|x,\theta)$)

$$r(\theta) = \frac{\theta_{1|1}(1-\theta_{1|1})}{\theta_{1|0}(1-\theta_{1|0})} \tag{47}$$

(the ratio of the variances of the two conditional distributions), such that this selection rule drives the ratio on the left hand side towards $r(\theta)$. The sampling of $x$'s is biased so that the more uncertain conditional distribution (having higher variance) is sampled more frequently.

**(2-4)** (20pts) The following matlab script `hw2_prob2.m` (with the "active" flag set to 1) runs the active learning procedure and displays parameter estimates and associated uncertainties. The results are stochastic (vary from run to run). The student is invited to run this script yourself (distributed with solutions).

```
% 'hw2_prob2.m'
% 6.867 Machine Learning
% Fall 2003

% initialization data
x = [0 0 0 0 0 1 1 1 1 1]';
y = [0 0 1 0 1 1 1 1 1 0]';
% initialize counters.
n0 = sum(x==0);
n1 = sum(x==1);
n01 = sum(x==0 & y==1);
n11 = sum(x==1 & y==1);

data = zeros(500,6);
active = 0;  % set to 0 for part (2-5)

% active learning loop...
for k = 1:500
  % calculate ML theta params
  theta0 = n01/n0;
  theta1 = n11/n1;
  % select x to sample next
  sigma0 = theta0 * (1-theta0);
  sigma1 = theta1 * (1-theta1);
  V0 = sigma0 / (n0*(n0+1));
  V1 = sigma1 / (n1*(n1+1));
  if (active)
    x = (V1 > V0);
  else
    x = (rand(1,1)>0.5); % stochastic sampling
  end
  % compute J's (for plots)
  J_est = sigma0/n0 + sigma1/n1;
  J_true = (.4*.6)/n0 + (.9*.1)/n1; % for comparison only
  % sample output for selected x
  y = query(x);
  % update counters
  if (x)
    n1=n1+1;
    if (y)
```

```
      n11=n11+1;
    end
  else % x==0
    n0=n0+1;
    if (y)
      n01=n01+1;
    end
  end
  % store data for generating plots
  data(k,1) = n0;
  data(k,2) = n1;
  data(k,3) = theta0;
  data(k,4) = theta1;
  data(k,5) = J_est;
  data(k,6) = J_true;
end

k = [1:500];

figure(1);
plot(k,data(k,3),'-',k,data(k,4),'--');
legend('theta0','theta1');
xlabel('iteration');
ylabel('estimate');
title('theta estimates');

figure(2);
plot(k,data(k,5),'--',k,data(k,6),'-');
legend('estimated','actual');
xlabel('iterations');
ylabel('J value');
title('overall variance');

figure(3);
n = k + 10;
plot(k,n.*data(k,5)','--',k,n.*data(k,6)','-');
legend('estimated','actual');
xlabel('iterations');
ylabel('n times J');
title('rescaled overall variance');
```

**(2-5)** (10pts) Rerun the script with the "active" flag set to 0. Observe that overall variance is typically larger (at the end of the run) under the stochastic sampling method. Averaging over many runs, the active learning "rate" $nJ$ converges to about .63 while the stochastic sampling converges to about .67. Hence, the active learning method, by biasing sampling

toward the more uncertain conditional distribution, does a better job on minimizing the overall variance of our estimates.

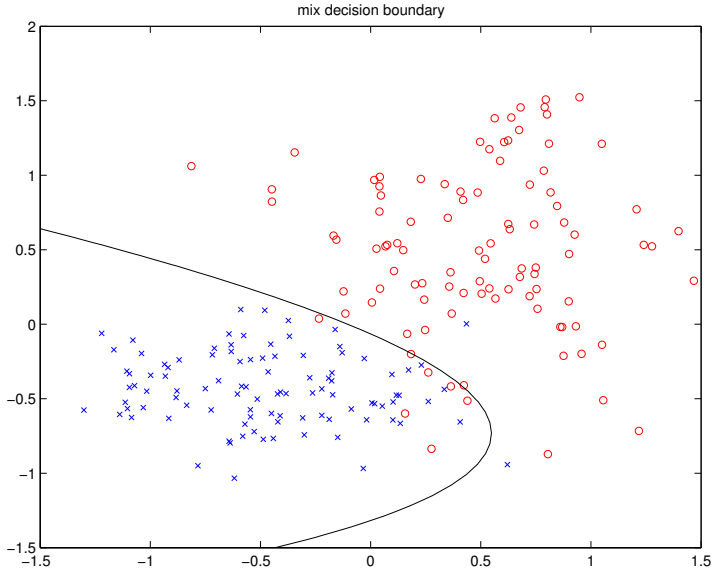
## Problem 3: generative and discriminative models

All MATLAB code for this problem is available in the script `hw2_prob3.m`. The script uses an additional function `loglogistic(z)` (not provided earlier) that simply evaluates $\log g(z)$ in a numerically stable manner. We will discuss here excerpts of the code and results. Please see/run the code for details.

**(3-1)** (10pts)

```
load data.mat;
n_train = size(trainD.y,1);
n_test = size(testD.y,1);

figure(1); plotdata(trainD); title('training data');
figure(2); plotdata(testD); title('test data');
mix = mixtrain(trainD.X,trainD.y);

% part a
% classify training data
train_loglik0 = mixloglik(trainD.X,zeros(size(trainD.y)),mix);
train_loglik1 = mixloglik(trainD.X,ones(size(trainD.y)),mix);
mix_train_err = mean(trainD.y ~= (train_loglik1 > train_loglik0))
% classify test data
test_loglik0 = mixloglik(testD.X,zeros(size(testD.y)),mix);
test_loglik1 = mixloglik(testD.X,ones(size(testD.y)),mix);
mix_test_err = mean(testD.y ~= (test_loglik1 > test_loglik0))
% part b
figure(3); mixboundary(mix,trainD); title('mix decision boundary');
% part c : compute the average conditional log-likelihoods for
% comparison with logistic models
mix_condloglik_train = (sum(mixloglik(trainD.X,trainD.y,mix)- ...
 log(exp(train_loglik0)+exp(train_loglik1))))/n_train
mix_condloglik_test = (sum(mixloglik(testD.X,testD.y,mix)- ...
 log(exp(test_loglik0)+exp(test_loglik1))))/n_test

mix_train_err =

    0.0450


mix_test_err =
```

```
    0.0490


mix_condloglik_train =

   -0.1003


mix_condloglik_test =

   -0.1170
```

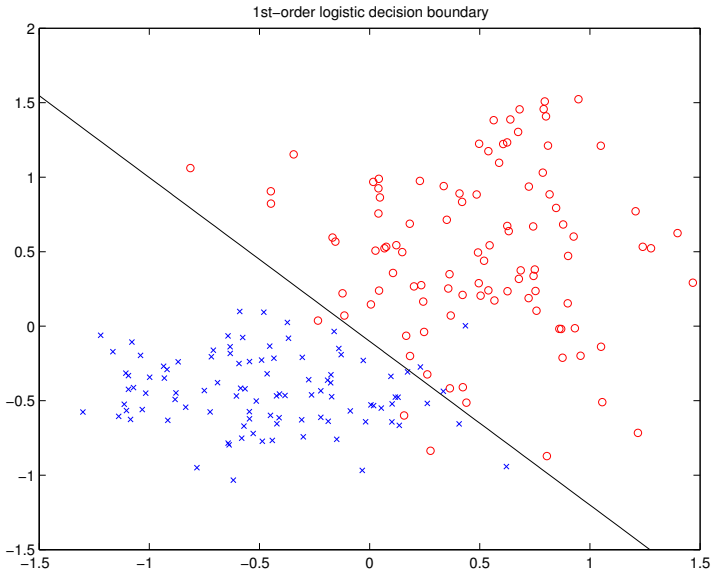



**(3-2)** (10pts)

```
% (3-2)
% part a
w = logisticreg(trainD.X,trainD.y)
input_train = trainD.X * w(2:length(w)) + w(1); % discriminant
input_test = testD.X * w(2:length(w)) + w(1);
logistic_train_err = mean(trainD.y ~= (input_train > 0))
logistic_test_err = mean(testD.y ~= (input_test > 0))

condloglik_train = (sum(loglogistic(input_train(find(trainD.y==1)))) + ...
    sum(loglogistic(-input_train(find(trainD.y==0)))))/n_train
% where we have used the fact that log(1-g(z)) = log g(-z)

condloglik_test = (sum(loglogistic(input_test(find(testD.y==1)))) + ...
```

```
    sum(loglogistic(-input_test(find(testD.y==0))))/n_test

% part b
figure(4); boundary(w,trainD); title('1st-order logistic decision boundary');

w =

   -0.7150
   -7.7597
   -7.0605


logistic_train_err =

    0.0300


logistic_test_err =

    0.0520


condloglik_train =

   -0.1063


condloglik_test =

   -0.1324
```

(3-2c) We would typically expect the discriminative method to achieve a higher conditional log-likelihood of the labels in the training set because this is precisely the criterion that the discriminative method is optimizing; the generative approach optimizes the joint log-likelihood of labels and examples and therefore deals with the conditional log-likelihood only indirectly. The results seem a bit contrary to this statement. The Gaussian mixture model performs better on the training set in terms of the conditional log-likelihood. The reason is that our comparison isn't fair. The Gaussian mixture model is more flexible, able to represent all quadratic decision boundaries, whereas the logistic regression model is restricted to linear boundaries.



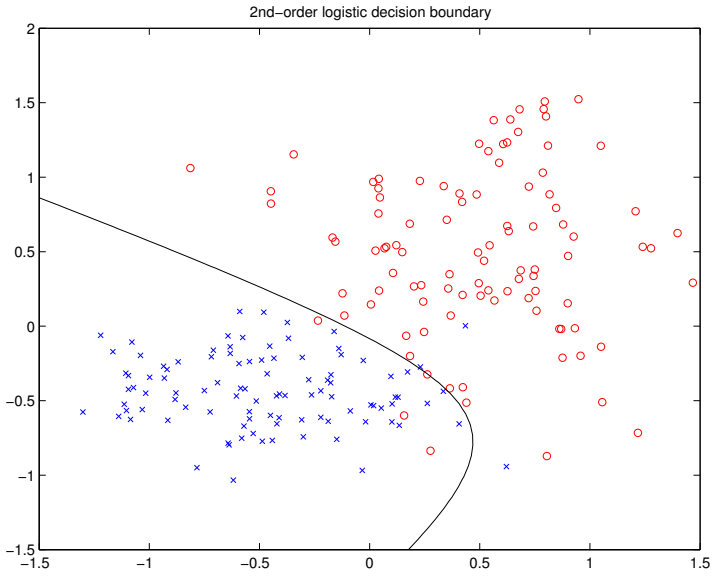

**(3-3)** (10pts)

```
phi = degexpand(trainD.X,2);
w = logisticreg(phi,trainD.y)
input_train = phi * w(2:length(w)) + w(1);
phi = degexpand(testD.X,2);
input_test = phi * w(2:length(w)) + w(1);
logistic_train_err2 = mean(trainD.y ~= (input_train > 0))
logistic_test_err2 = mean(testD.y ~= (input_test > 0))

condloglik_train2 = (sum(loglogistic(input_train(find(trainD.y==1)))) + ...
    sum(loglogistic(-input_train(find(trainD.y==0)))))/n_train
condloglik_test2 = (sum(loglogistic(input_test(find(testD.y==1)))) + ...
    sum(loglogistic(-input_test(find(testD.y==0)))))/n_test

figure(5); boundary(w,trainD,2); title('2nd-order logistic decision boundary');


w =

   -0.8767
   -7.5253
  -11.0026
    3.6968
    1.8570
    1.8570
   -5.9708
```

```
logistic_train_err2 =

    0.0350


logistic_test_err2 =

    0.0500


condloglik_train2 =

   -0.0950


condloglik_test2 =

   -0.1221
```

The quadratic logistic regression model has the highest conditional log-likelihood of the training labels, as expected. It can represent the same boundaries as the Gaussian mixture model but it is trained discriminatively. This is indeed what should always happen (without regularization). Note that the linear logistic model achieves a lower classification error in the training set. This is possible since we are not optimizing the classification error directly. The two measures are related but they are not identical.

The Gaussian mixture model seems to generalize the best, both in terms of classification error and the conditional log-likelihood of the test labels. This is because the data points were indeed generated from such a Gaussian mixture model.




**(3-4)** (10pts)

We first note that a second order (quadratic) function of a vector $x \in \Re^d$ can be written as $x'Ax + b'x + c$ where $A \in \Re^{d\times d}$ is a symmetric (but not necessarily positive definite) matrix, $b \in \Re^d$ is vector, and $c \in \Re$ is a scalar. Hence, logistic 2nd order models are given by $\Pr(Y = 1|x) = \frac{1}{1+e^{-(x'Ax+b'x+c)}}$ for any such $A$, $b$ and $c$.

Now, for a generative Gaussian mixture model with $\Pr(y) = p_y$ and

$$\Pr(x|y) = \frac{1}{\sqrt{2\pi|\Sigma_y|}} e^{-\frac{1}{2}(x-\mu_y)'\Sigma_y^{-1}(x-\mu_y)} \tag{48}$$

we have:

$$\begin{aligned}
\Pr(Y = 1|x) &= \frac{\Pr(Y = 1)\Pr(x|Y = 1)}{\Pr(Y = 1)\Pr(x|Y = 1) + \Pr(Y = 0)\Pr(x|Y = 0)} && (49)\\
&= \frac{p_1 \frac{1}{\sqrt{2\pi|\Sigma_1|}} e^{-\frac{1}{2}(x-\mu_1)'\Sigma_1^{-1}(x-\mu_1)}}{p_1 \frac{1}{\sqrt{2\pi|\Sigma_1|}} e^{-\frac{1}{2}(x-\mu_1)'\Sigma_1^{-1}(x-\mu_1)} + p_0 \frac{1}{\sqrt{2\pi|\Sigma_0|}} e^{-\frac{1}{2}(x-\mu_0)'\Sigma_0^{-1}(x-\mu_0)}} && (50)\\
&= \frac{1}{1 + \frac{p_1}{1-p_1} e^{-\frac{1}{2}((x-\mu_0)'\Sigma_0^{-1}(x-\mu_0)-(x-\mu_1)'\Sigma_1^{-1}(x-\mu_1)-|\Sigma_0|+|\Sigma_1|)}} && (51)\\
&= \frac{1}{1 + e^{-\frac{1}{2}((x-\mu_0)'A_0(x-\mu_0)-(x-\mu_1)'A_1(x-\mu_1)+|A_0|-|A_1|)+r}} && (52)\\
&= \frac{1}{1 + e^{-\frac{1}{2}x'(A_0-A_1)x+(\mu_0'A_0-\mu_1'A_1)x-\frac{1}{2}(\mu_0'A\mu_0-\mu_1'A\mu_1'+|A_0|-|A_1|)+r}} && (53)\\
&= \frac{1}{1 + e^{-(x'Ax+b'x+c)}} && (54)
\end{aligned}$$

where $A_y = \Sigma_y^{-1}$ and $r = \log \frac{p_1}{1-p_1}$ and

- $A = \frac{1}{2}(A_0 - A_1) = \frac{1}{2}(\Sigma_0^{-1} - \Sigma_1^{-1})$ is a symmetric matrix (since $A_0, A_1$ are both symmetric, being inverses of symmetric matrices),
- $b' = -(\mu_0' A_0 - \mu_1' A_1) = -(\mu_0' \Sigma_0^{-1} - \mu_1' \Sigma_1^{-1})$
- and $c = \frac{1}{2}(\mu_0' A \mu_0 - \mu_1' A \mu_1' + |A_0| - |A_1|) - r = \frac{1}{2}(\mu_0' \Sigma_0^{-1} \mu_0 - \mu_1' \Sigma_1^{-1} \mu_1' - |\Sigma_0| + |\Sigma_1|) - \log \frac{p_1}{1-p_1}$.

This establishes that the conditional distribution $\Pr(y|x)$ of a Gaussian mixture model is a second order logistic. To establish that every second order logistic model can be interpreted as the conditional distribution of a Gaussian mixture model, we must find, for each $A, b, c$, suitable $p_1 \in [0,1]$, $\mu_0, \mu_1 \in \Re^d$ and symmetric positive definite $\Sigma_0, \Sigma_1 \in \Re^{d \times d}$ such that plugging these in above, we would recover our desired $A, b, c$.

Given $A, b, c$ we first determine the inverse covariances $A_0$ and $A_1$. These should be selected such that half their difference is equal to $A$. The difficult constrain here is that $A_0, A_1$ need to be positive definite. If $A$ is already positive definite, this is not a problem, and we can select, for example, $A_0 = 2A$ and $A_1 = A$. If $A$ is not positive definite, let $-\lambda \leq 0$ be the smallest eigenvalue of $A$, then $A + (1+\lambda)I$ is positive definite (since we are adding $1+\lambda$ to all eigenvalues, hence making them all positive). Selecting $A_0 = 2(A + (1+\lambda)I)$ and $A_1 = 2(1+\lambda)I$, both positive definite, yields $A = \frac{1}{2}(A_0 - A_1)$. The covariance matrices will be $\Sigma_0 = A_0^{-1}$ and $\Sigma_1 = A_1^{-1}$.

After determining the covariance matrices, we next set the means of the two Gaussians according to $b$. We can always set $\mu_0 = 0$ and $\mu_1 = \Sigma_1 b$ yielding $-(\mu_0' \Sigma_0^{-1} - \mu_1' \Sigma_1^{-1}) = -(0\Sigma_0^{-1} - b'\Sigma_1\Sigma_1^{-1}) = b'$.

To complete the correspondence, we need to set $r = \frac{1}{2}(\mu_0' A \mu_0 - \mu_1' A \mu_1' + |A_0| - |A_1|) - c$ by inverting $r(p_1) = \log \frac{p_1}{1-p_1}$. This is possible since this transformation maps values between zero and one (the possible values of the probability $p_1$) to the entire range of real numbers, with $r(0) = \log 0 = -\infty$ and $r(1) = \log \infty = \infty$.

Note that we had great freedom in selecting the covariance matrices and means, each time having twice as many parameters as constraints. There are many possible Gaussian mixture models that yield the same (second order logistic) conditional distribution.

**(3-5)** (10pts)

As we proved above, the conditional distribution $\Pr(y|x)$ resulting from a Gaussian mixture model is itself a possible second order logistic model. The conditional log-likelihood of the Gaussian mixture model depends only on this conditional distribution. But when we train a logsitic model, we specifically maximize the conditional log-likelihood, hence the trained logistic model will have the maximal log-likelihood of all possible second order logistic models, and it cannot be lower than the conditional log-likelihood of the trained Gaussian mixture model (since this conditional log-likelihood is also in the class of second order logsitic models).

Note that when training a Gaussian mixture model, we maximize the *joint* likelihood rather then the *conditional* likelihood, hence although the trained Gaussian mixture model is guranteed to have maximal joint likelihood, it is not guranteed to have maximal conditional likelihood (and will usually not have maximal conditional likelihood).

**(3-6)** (10pts)

```
for k = 1:4
  phi = degexpand(trainD.X,k);
  w = logisticreg(phi,trainD.y);
  input_train = phi * w(2:length(w)) + w(1);

  condloglik_train = (sum(loglogistic(input_train(find(trainD.y==1)))) + ...
      sum(loglogistic(-input_train(find(trainD.y==0)))))/n_train;

  phi = degexpand(testD.X,k);
  input_test = phi * w(2:length(w)) + w(1);

  condloglik_test = (sum(loglogistic(input_test(find(testD.y==1)))) + ...
    sum(loglogistic(-input_test(find(testD.y==0)))))/n_test;

  fprintf('deg: %d, train log-lik: %f, ave test log-lik: %f\n',k, ...
  condloglik_train,condloglik_test);
end
```

```
deg: 1, train log-lik: -0.106334, ave test log-lik: -0.132366
deg: 2, train log-lik: -0.095050, ave test log-lik: -0.122121
deg: 3, train log-lik: -0.060270, ave test log-lik: -0.267130
deg: 4, train log-lik: -0.053588, ave test log-lik: -0.398441
```

We naturally get more complex classifiers as we increase the degree of the polynomial features. The resulting sets of classifiers are nested in the sense that any classifier corresponding to degree $m-1$ features is contained in the set of classifiers we can represent with degree $m$ features. Because of this nested organization, the structural error must go down (or at least not increase) as a function of the degree. However, the approximation error will necessarily go up since it will be harder to find the classifier that generalizes the best when the set of possibilities is larger but the number of training examples remains the same.

Let's see how we can use these ideas to explain the results. First, the training log-likelihood has to increase as a function of degree since the classes are nested. Second, our results indicate that the test conditional log-likelihood first increases and then decreases rapidly. The increase has to come from reducing structural error (lower structural error means higher likelihood) since the approximation error can only increase (and therefore would decrease the likelihood). After degree 2 the error is likely to be dominated by the approximation error.

We know that degree 2 gives the correct model, since we generated the data. In this case the structural errors corresponding to degree 3 and 4 features are exactly the same as for degree 2 (the correct answer). We cannot do better than the correct answer in terms of structural error. So the rapid decline of the log-likelihood after degree 2 is indeed due to only the approximation error.