EEC445: Introduction to Machine Learning
Homework #2 Solutions

Notation: As in the lecture, we will interchangeably use $\mathbf{x}_i$ and $\mathbf{x}^{(i)}$ to denote the i-th training example (and similar notation for other variables). Often, it's less confusing to use $\mathbf{x}^{(i)}$ than $\mathbf{x}_i$ (especially in handwriting) when your expression involves another subscripts to denote specific coordinates in the vector. The Bishop's book uses the former notation, but some other machine learning textbooks use the latter notation. You are free to use whichever notation as long as you are consistent.

1 [25 points] Gaussian Discriminate Analysis

(a) **Answer[8 points]:** Since the given formulae are conditioned on y, use Bayes rule to get:

$$\begin{aligned}
p(y=1|\mathbf{x};\phi,\Sigma,\mu_0,\mu_1) &= \frac{p(\mathbf{x}|y=1;\phi,\Sigma,\mu_0,\mu_1)p(y=1;\phi,\Sigma,\mu_0,\mu_1)}{p(\mathbf{x};\phi,\Sigma,\mu_0,\mu_1)} \\
&= \frac{p(\mathbf{x}|y=1;\dots)p(y=1;\dots)}{p(\mathbf{x}|y=1;\dots)p(y=1;\dots)+p(\mathbf{x}|y=0;\dots)p(y=0;\dots)} \\
&= \frac{\exp(-\frac{1}{2}(\mathbf{x}-\mu_1)^T\Sigma^{-1}(\mathbf{x}-\mu_1))\phi}{\exp(-\frac{1}{2}(\mathbf{x}-\mu_1)^T\Sigma^{-1}(\mathbf{x}-\mu_1))\phi+\exp(-\frac{1}{2}(\mathbf{x}-\mu_0)^T\Sigma^{-1}(\mathbf{x}-\mu_0))(1-\phi)} \\
&= \frac{1}{1+\frac{1-\phi}{\phi}\exp(-\frac{1}{2}(\mathbf{x}-\mu_0)^T\Sigma^{-1}(\mathbf{x}-\mu_0)+\frac{1}{2}(\mathbf{x}-\mu_1)^T\Sigma^{-1}(\mathbf{x}-\mu_1))} \\
&= \frac{1}{1+\exp(\log(\frac{1-\phi}{\phi})-\frac{1}{2}(\mathbf{x}-\mu_0)^T\Sigma^{-1}(\mathbf{x}-\mu_0)+\frac{1}{2}(\mathbf{x}-\mu_1)^T\Sigma^{-1}(\mathbf{x}-\mu_1))}
\end{aligned}$$

In this expression, the quadratic terms $\mathbf{x}^T\Sigma^{-1}\mathbf{x}$ cancel, and only terms linear in x are left over. It is clear that this expression can be rewritten in the form $\frac{1}{1+\exp(-\mathbf{w}^T\mathbf{x})}$after adding a constant intercept term $x_0 = 1$.

(b) **[17 (+ 7 extra credit) points]** Here we solve for (c), since (b) is just a special case of c.

First, derive the expression for the log-likelihood of the training data:

$$\begin{aligned}
\ell(\phi,\mu_0,\mu_1,\Sigma) &= \log\prod_{i=1}^{N}p(\mathbf{x}_i|t_{i;};\mu_0,\mu_1,\Sigma)p(t_i;\phi) \\
&= \sum_{i=1}^{N}\log p(\mathbf{x}_i|t_{i;};\mu_0,\mu_1,\Sigma)+\sum_{i=1}^{N}\log p(t_i;\phi) \\
&\approxeq \sum_{i=1}^{N}[\frac{1}{2}\log\frac{1}{|\Sigma|}-\frac{1}{2}(\mathbf{x}_i-\mu_{t_i})^T\Sigma^{-1}(\mathbf{x}_i-\mu_{t_i})+t_i\log\phi+(1-t_i)\log(1-\phi)]
\end{aligned}$$

where constant terms independent of the parameters have been ignored in the last expression. Now, the likelihood is maximized by setting the derivative (or gradient) with respect to each of the parameters to zero.

$$\begin{aligned}\frac{\partial \ell}{\partial \phi} &= \sum_{i=1}^{N}[\frac{t_i}{\phi} - \frac{1-t_i}{1-\phi}] \\ &= \frac{\sum_{i=1}^{N} 1\{t_i = 1\}}{\phi} - \frac{N - \sum_{i=1}^{N} 1\{1-t_i\}}{1-\phi}]\end{aligned}$$

Setting this equal to zero and solving for ϕgives the maximum likelihood estimate. For μ_0, take the gradient of the log-likelihood, and then use the same kinds of tricks as were used to analytically solve the linear regression problem.

$$\begin{aligned}\nabla_{\mu_0}\ell &= -\frac{1}{2}\sum_{i:t_i=0} \nabla_{\mu_0}(\mathbf{x}_i - \mu_0)^T\Sigma^{-1}(\mathbf{x}_i - \mu_0) \\ &= -\frac{1}{2}\sum_{i:t_i=0} \nabla_{\mu_0}[\mu_0^T\Sigma^{-1}\mu_0 - \mathbf{x}_i^T\Sigma^{-1}\mu_0 - \mu_0^T\Sigma^{-1}\mathbf{x}_i] \\ &= -\frac{1}{2}\sum_{i:t_i=0} \nabla_{\mu_0} tr[\mu_0^T\Sigma^{-1}\mu_0 - \mathbf{x}_i^T\Sigma^{-1}\mu_0 - \mu_0^T\Sigma^{-1}\mathbf{x}_i] \\ &= -\frac{1}{2}\sum_{i:t_i=0} [2\Sigma^{-1}\mu_0 - 2\Sigma^{-1}\mathbf{x}_i]\end{aligned}$$

The last step uses matrix calculus identities (specifically, those given in page 8 of the lecture notes), and also the fact that Σ(and thus Σ^{-1}) is symmetric. Setting this gradient to zero gives the maximum likelihood estimate for μ_0. The derivation for $\mu_1$is similar to the one above.

For Σ, we find the gradient with respect to $S = \Sigma^{-1}$ rather than just to simplify the derivation (note that $|S| = \frac{1}{|\Sigma|}$). You should convince yourself that the maximum likelihood estimate S_{ML} found in this way would correspond to the actual maximum likelihood estimate Σ_{ML} as $S_{ML}^{-1} = \Sigma_{ML}$. Additionally, we let $\mathbf{b}_i = (\mathbf{x}_i - \mu_{t_i})$. This gives us:

$$\begin{aligned}\nabla_S\ell &= \sum_{i=1}^{N} \nabla_S[\frac{1}{2}\log|S| - \frac{1}{2}\mathbf{b}_i^T S\mathbf{b}_i] \\ &= \sum_{i=1}^{N}[\frac{1}{2|S|}\nabla_S|S| - \frac{1}{2}\nabla_S\mathbf{b}_i^T S\mathbf{b}_i]\end{aligned}$$

But, we have the following identities:

$$\nabla_S|S| = |S|(S^{-1})^T$$

$$\nabla_S\mathbf{b}_i^T S\mathbf{b}_i = \nabla_S tr(\mathbf{b}_i^T S\mathbf{b}_i) = \nabla_S tr(S\mathbf{b}_i\mathbf{b}_i^T) = \mathbf{b}_i\mathbf{b}_i^T$$

In the above, we again used matrix calculus identities, and also the commutatitivity of the trace operator for square matrices. Putting these into the original equation, we get:

$$\begin{aligned}\nabla_S \ell &= \sum_{i=1}^{N}[\frac{1}{2}S^{-1} - \frac{1}{2}\mathbf{b}_i\mathbf{b}_i^T] \\ &= \frac{1}{2}\sum_{i=1}^{N}[\Sigma - \mathbf{b}_i\mathbf{b}_i^T]\end{aligned}$$

Setting this to zero gives the required maximum likelihood estimate for Σ.

(c) Estimated parameter values:

$\phi = 0.43$
$\mu_0 = [1.1731, 1.7420]$
$\mu_1 = [-0.0914, -0.6088]$
$\Sigma = [0.6570, 0.6878; 0.6878, 0.8956]$

Test accuracy: 0.92.

2 [35 points] Softmax Regression via Gradient Ascent

(a) **[18 points]** We have:

$$l(\mathbf{w}) = \sum_{i=1}^{N}\sum_{k=1}^{K}\log\left[p(t_i = k|\mathbf{x}_i; \mathbf{w})\right]^{I(t_i=k)}$$

Then:

$$\begin{aligned}
\nabla_{\mathbf{w}_m} l(\mathbf{w}) &= \nabla_{\mathbf{w}_m} \sum_{i=1}^{N}\sum_{k=1}^{K} \log\left(p(t_i = k|\mathbf{x}_i;\mathbf{w})\right)^{I(t_i=k)} \\
&= \nabla_{\mathbf{w}_m} \sum_{i=1}^{N}\sum_{k=1}^{K} I(t_i = k)\log\left(p(t_i = k|\mathbf{x}_i;\mathbf{w})\right) \\
&= \sum_{i=1}^{N} \nabla_{\mathbf{w}_m} \sum_{k=1}^{K} I(t_i = k)\left[\log\left(p(t_i = k|\mathbf{x}_i;\mathbf{w})\right)\right] \\
&= \sum_{i=1}^{N} \nabla_{\mathbf{w}_m} \sum_{k=1}^{K} I(t_i = k)\left[\log\left(\frac{\exp\left(\mathbf{w}_k^T\phi(\mathbf{x}_i)\right)}{1+\sum_{j=1}^{K-1}\exp\left(\mathbf{w}_j^T\phi(\mathbf{x}_i)\right)}\right)\right] \\
&= \sum_{i=1}^{N} \nabla_{\mathbf{w}_m} \sum_{k=1}^{K} I(t_i = k)\left[\log\left(\exp\left(\mathbf{w}_k^T\phi(\mathbf{x}_i)\right)\right) - \log\left(1+\sum_{j=1}^{K-1}\exp\left(\mathbf{w}_j^T\phi(\mathbf{x}_i)\right)\right)\right] \\
&= \sum_{i=1}^{N} \nabla_{\mathbf{w}_m} \left(\sum_{k=1}^{K} I(t_i = k)\left[\mathbf{w}_k^T\phi(\mathbf{x}_i) - \log\left(1+\sum_{j=1}^{K-1}\exp\left(\mathbf{w}_j^T\phi(\mathbf{x}_i)\right)\right)\right]\right)
\end{aligned}$$

The log term inside summation does not contain k:

$$= \sum_{i=1}^{N} \nabla_{\mathbf{w}_m} \left(\left[\sum_{k=1}^{K} I(t_i = k)\mathbf{w}_k^T\phi(\mathbf{x}_i)\right] - \left[\sum_{k=1}^{K} I(t_i = k)\log\left(1+\sum_{j=1}^{K-1}\exp\left(\mathbf{w}_j^T\phi(\mathbf{x}_i)\right)\right)\right]\right)$$

The right summation only has one none-zero term:

$$\begin{aligned}
&= \sum_{i=1}^{N} \nabla_{\mathbf{w}_m} \left(\left[\sum_{k=1}^{K} I(t_i = k)\mathbf{w}_k^T\phi(\mathbf{x}_i)\right] - \log\left(1+\sum_{j=1}^{K-1}\exp\left(\mathbf{w}_j^T\phi(\mathbf{x}_i)\right)\right)\right) \\
&= \sum_{i=1}^{N} \nabla_{\mathbf{w}_m} \left[\sum_{k=1}^{K} I(t_i = k)\mathbf{w}_k^T\phi(\mathbf{x}_i)\right] - \nabla_{\mathbf{w}_m} \log\left(1+\sum_{j=1}^{K-1}\exp\left(\mathbf{w}_j^T\phi(\mathbf{x}_i)\right)\right)
\end{aligned}$$

The left term contains $\mathbf{w}_m$ iff $m = t_i$. Second term contains $\mathbf{w}_m$ (produced by summation)

$$\begin{aligned}
&= \sum_{i=1}^{N} I(t_i = m)\phi(\mathbf{x}_i) - \frac{1}{1+\sum_{j=1}^{K-1}\exp\left(\mathbf{w}_j^T\phi(\mathbf{x}_i)\right)} \cdot \exp\left(\mathbf{w}_m^T\phi(\mathbf{x}_i)\right)\cdot\phi(\mathbf{x}_i) \\
&= \sum_{i=1}^{N} \phi(\mathbf{x}_i)\left[I(t_i = m) - p(t_i = m|\mathbf{x}_i)\right] \qquad (1)
\end{aligned}$$

(b) **[17 points]** Accuracy should be 92% (error = 8%, miss-classifying 4 examples), identical to mnrfit.

(Testing) code for computing accuracy (possible filename `q2_compute_accuracy.m`):

```
% Compute Accuracy:
correct = 0;
for i=101:150
    probs = compute_softmax_probs(W, F(i, :)');
    [m, idx] = max(probs);
    if (idx == labels(i))
        correct = correct + 1;
    end
```

```
end

accuracy = correct / (150 - 101 + 1)
```

(Training) - fitting the parameters:

```
num_labels = max(q2t_test);
W = zeros(num_labels, size(F, 2));

alpha = 0.0005;
for j = 1:400
    % A single iteration over all training examples
    delta_w = zeros(num_labels, size(F,2));
    for i = 1:100  %size(F,1)
        indicator = zeros(1, num_labels);
        indicator(labels(i)) = 1;
        probs = compute_softmax_probs(W, F(i, :)');

        delta_w = delta_w + (F(i, :)' * ( indicator - probs ))';
    end

    o = W +  alpha * (delta_w);
    W(1:(num_labels - 1), :) = o(1:(num_labels - 1), :);
    compute_accuracy
end
```

Helper function

```
function [ probs ] = compute_softmax_probs( W, x )
    % This function computes probabilities for x belonging
    % K classes. W is a matrix of size (K-1) * M. x is M
    probs = softmax(double(W*x))';
    return;
end
```

3 [40 points] Naive Bayes for classifying SPAM

(a) **[15 points]** The test error when training on the full training set was 1.625%. If you got a different error (or if you got the words website and lowest for part b), you most probably implemented the wrong Naive Bayes Model.

(b) **[10 points]** The five most indicative words for the spam class were: `httpaddr`, `spam`, `unsubscrib`, `ebai` and `valet`.

(c) **[15 points]** The test error for different training set sizes was:

- Training set size 50: Test set error = 3.875%
- Training set size 100: Test set error = 2.625%
- Training set size 200: Test set error = 2.625%
- Training set size 400: Test set error = 1.875%
- Training set size 800: Test set error = 1.75%
- Training set size 1400: Test set error = 1.625%