CS 189 Introduction to Machine Learning
Spring 2018 gs2

# 1 Descent

(a) Consider the setup of gradient descent: $x_{t+1} = x_t - \alpha \nabla f(x)$. Recall that for a function $f : \mathbb{R}^n \to \mathbb{R}$ that is convex and differentiable, and which is additionally L-smooth:
$f(y) \leq f(x) + \nabla f(x)^T (y-x) + \frac{L}{2}\|x-y\|_2^2$ for any $x, y$.
Then if we run gradient descent for $k$ iterations with a fixed step size $\alpha = 1/L$, it will yield a solution $f(x^{(k)})$ which satisfies:

$f(x^{(k)}) - f(x^*) \leq \frac{\|x^{(0)} - x^*\|_2^2}{2\alpha k}$

What is the convergence rate? Hint: how many iterations are required to obtain $f(x^{(k)}) - f(x^*) \leq \varepsilon$?

**Solution:** $\varepsilon \geq \frac{\|x^{(0)} - x^*\|_2^2}{2\alpha k}$
Can be rewritten as: $k \geq \frac{\|x^{(0)} - x^*\|_2^2}{2\alpha} \cdot \frac{1}{\varepsilon}$
Since $\frac{\|x^{(0)} - x^*\|_2^2}{2\alpha}$ is a constant term, this implies that $O(\frac{1}{\varepsilon})$ iterations are required to converge.
Ie, the convergence rate is $O(\frac{1}{\varepsilon})$.

(b) Now, consider the case of strong convexity, that is:
$f(y) \geq f(x) + \nabla f(x)^T (y-x) + \frac{\mu}{2}\|x-y\|_2^2$

How does assuming strong convexity impact the convergence rate? That is, how many iterations are required to obtain a value of x that is $\varepsilon$ away from $x^*$ if we run gradient descent for k iterations with a constant stepsize $\alpha = \frac{1}{L}$?
Hint: For $x_{t+1}$, the norm squared error is bounded by: $\|x_{t+1} - x^*\|_2^2 \leq (1 - \frac{\mu}{L})\|x_t - x^*\|_2^2$

**Solution:** By induction, after k iterations:
$\|x_k - x^*\|_2^2 \leq (1 - \frac{\mu}{L})^k \|x_0 - x^*\|_2^2$
To bound error $\|x_k - x^*\|_2^2 \leq \varepsilon$: $\varepsilon \leq (1 - \frac{\mu}{L})^k \|x_0 - x^*\|_2^2$
Take the log of both sides: $\log(\varepsilon) \leq k \log(1 - \frac{\mu}{L}) + \log(\|x_0 - x^*\|_2^2)$
Which can be rewritten as: $k \geq \frac{1}{\log(1-\frac{\mu}{L})} \log(\frac{\varepsilon}{\|x_0 - x^*\|_2^2})$
Which is equivalent to $k \geq \frac{-1}{\log(1-\frac{\mu}{L})} \log(\frac{\|x_0 - x^*\|_2^2}{\varepsilon})$
Since $\log(1 - \frac{\mu}{L}) < 0$ and $\|x_0 - x^*\|_2^2$ is a constant term: This implies that $O(\log(\frac{1}{\varepsilon}))$ iterations are required. This is an improvement from the convex case for small $\varepsilon$.

Proof that $\|x_{t+1}-x^*\|_2^2 \leq (1-\frac{\mu}{L})\|x_t-x^*\|_2^2$:
$\|x_{t+1}-x^*\|_2^2 = \|x_t - \frac{1}{L}\nabla f(x_t) - x^*\|_2^2$
$= \|x_t-x^*\|_2^2 - \frac{2}{L}\nabla f(x_t)^T(x_t-x^*) + \frac{1}{L^2}\|\nabla f(x_t)\|_2^2$
Plugging in $\|\nabla f(x_t)\|_2^2 \leq 2L(f(x_t)-f(x^*))$:
$\|x_{t+1}-x^*\|_2^2 \leq \|x_t-x^*\|_2^2 - \frac{2}{L}\nabla f(x_t)^T(x_t-x^*) + \frac{2}{L}(f(x_t)-f(x^*))$
Now, rewriting the definition of strong convexity as:
$\nabla f(x)^T(x-x^*) \geq f(x)-f(x^*)+\frac{\mu}{2}\|x-x^*\|_2^2$
Substituting in for $\nabla f(x)^T(x-x^*)$ will give us:
$\|x_{t+1}-x^*\|_2^2 \leq \|x_t-x^*\|_2^2 - \frac{2}{L}(f(x)-f(x^*)+\frac{\mu}{2}\|x-x^*\|_2^2) + \frac{2}{L}(f(x_t)-f(x^*))$
Grouping like terms:
$\|x_k-x^*\|_2^2 \leq (1-\frac{\mu}{L})\|x_t-x^*\|_2^2 - \frac{2}{L}(f(x)-f(x^*)) + \frac{2}{L}(f(x_t)-f(x^*))$
Which implies that $\|x_{t+1}-x^*\|_2^2 \leq (1-\frac{\mu}{L})\|x_t-x^*\|_2^2$

(c) Above, we have seen how Gradient Descent works for certain functions. However,evaluating the gradient on all data points is expensive. This is one reason we use Stochastic Gradient Descent in practice. Specifically, recall that in SGD instead of using the full gradient, we use a noisy estimate of it. Specifically, we assume:

$$\tilde{g}(x) = \nabla(x) + \varepsilon \sim N(0,\Sigma)$$

We will now explore the role of this noise. For concreteness let $f(w) = \frac{1}{2}||Xw-y||_2^2$ i.e. we are trying to optimize the OLS loss function using SGD.

Recall that in pratice we usually choose $\eta$ to be small (Discuss with the people around you why we want this. One reason can be see by considering what happens to the norm of the gradient as we approach the optimal value). That being said, there are also reasons to want high learning rates. To see this, consider the extreme case of $\eta = 0$. Then, we have $w^1 = w^0 - 0 = w^0$ and thus we never advance from our initial case. Formally show a stonger version of this by proving a lower bound on $\eta$ if we want to make at least $l$ progress on the $k$th step i.e. we want $||w^{k+1}-w^k||_2^2 \geq l$
HINT: If we prove an upper bound on $||w^{k+1}-w^k||_2^2$, we can then lower bound that upper bound with $l$.
**Solution:** First we use the definition of SGD:

$$w^{k+1} = w^k - \eta\tilde{g}(w^k) = w^k - \eta(\nabla(w^k)+\varepsilon) = w^k - \eta(X^TXw^k - X^Ty + \varepsilon)$$

$$w^k - w^{k+1} = \eta(X^TXw^k - X^Ty + \varepsilon)$$

$$||w^k - w^{k+1}||_2^2 = \eta||X^TXw^k - X^Ty + \varepsilon||_2^2$$

$$||w^{k+1} - w^|||_2^2 \leq \eta(||X^TXw^k||_2^2 + ||X^Ty||_2^2 + ||\varepsilon||_2^2)$$

By the Triangle Inequality. Since we want the norm of the progress made to be at least $l$, we also need:

$$\eta(||X^TXw^k||_2^2 + ||X^Ty||_2^2 + ||\varepsilon||_2^2) \geq l$$

Now, note that while $\varepsilon$ is zero-mean, the norm of $\varepsilon$ is not. In fact, as you should prove to yourself, $E[||\varepsilon||_2^2] = tr(\Sigma)$. Since the trace is positive (because the covariance matrix is assumed to be non-degenerate i.e. positive) we have that SGD always requires a larger learning rate than GD to make the same progress at each step. But, this is in conflict with the desire above for a small learning rate. Thus, we see the inherent conflict in SGD and why choosing a good step size is difficult.

## 2 Backprop: *Adapted from Fall 2015 Final Exam*

Suppose we have a neural network that takes in $P = 2$ input features, has $H = 4$ hidden units, and $N = 1$ output.

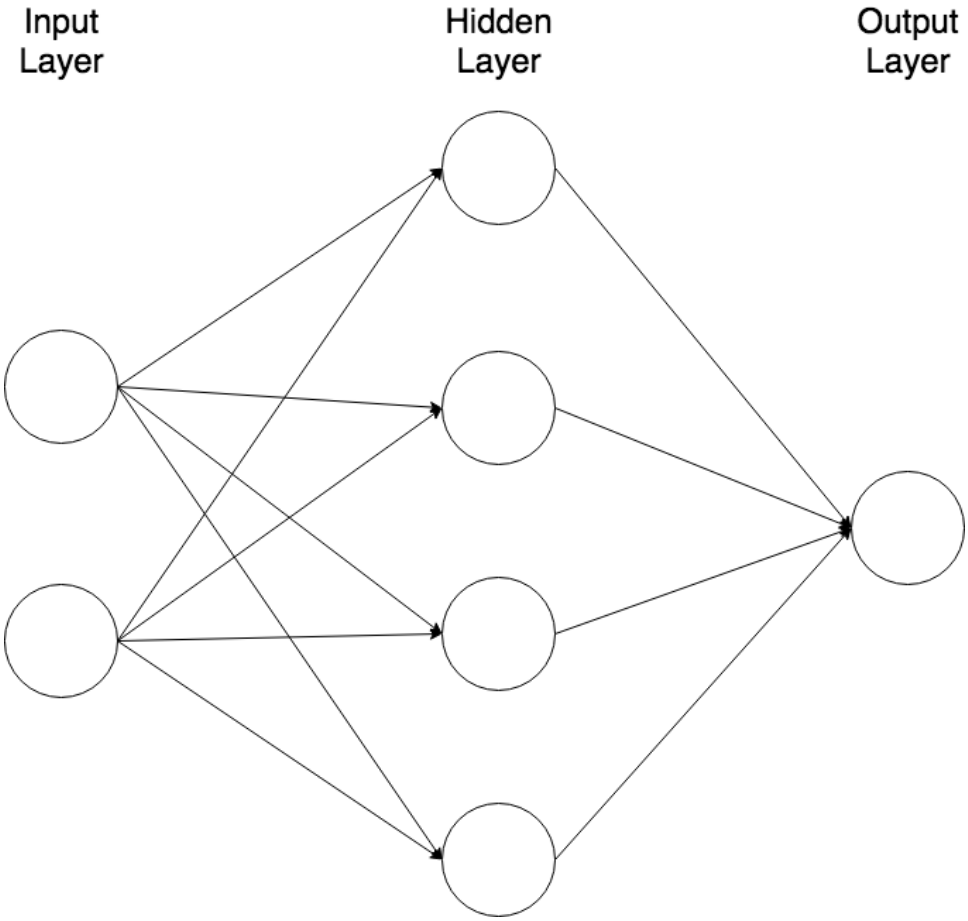



Figure 1: Unlabeled neural network as described above.

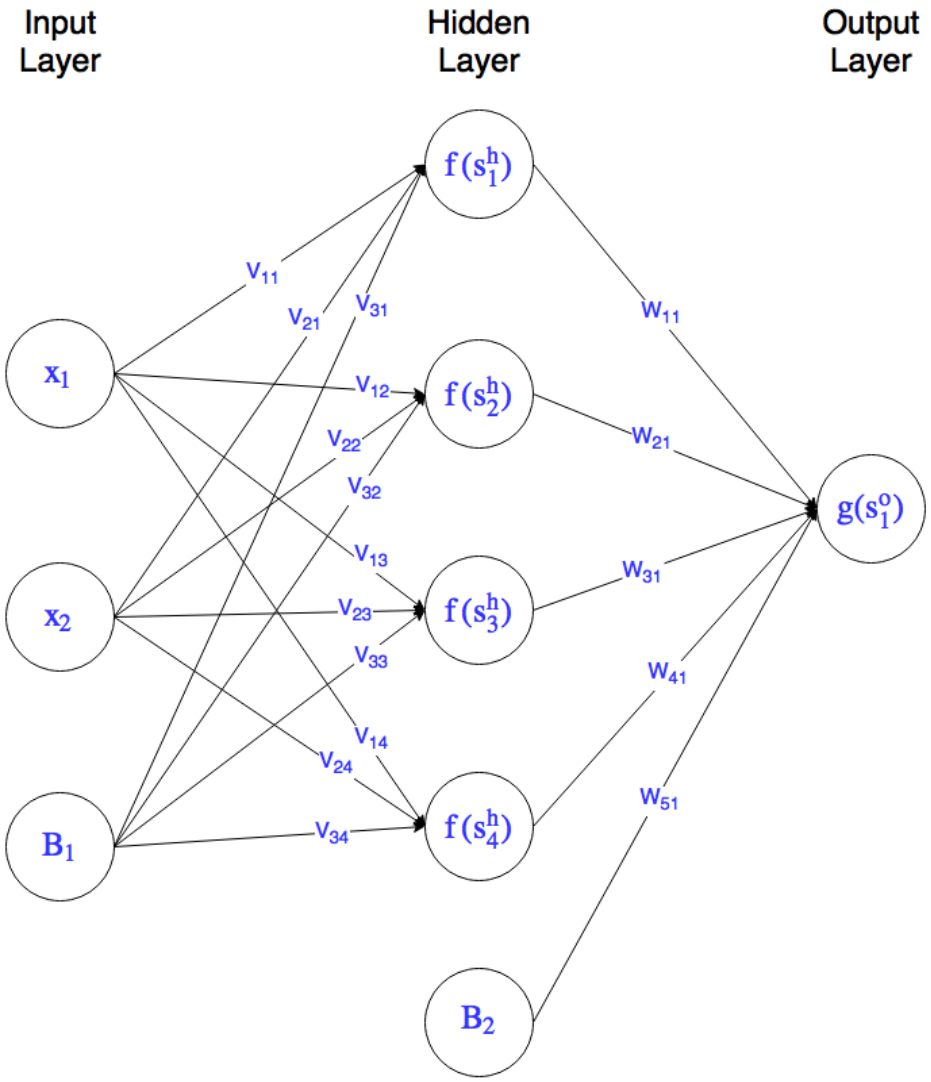
$f(x)$ is the activation function in the hidden layer, and $g(x)$ is the activation function in the output layer.

Let's define some notation!
$x \in \mathscr{R}^P$ is a feature vector
$s_j^h = \sum_{i=1}^{P} V_{i,j} x_i$ are inputs to the hidden layer
$h_j = f(s_j^h)$
$s_k^o = \sum_{j=1}^{H} W_{j,k} h_j$ are inputs to the output layer
$z_k = g(s_k^o)$
$V \in \mathscr{R}^{H \times P}$ are weights between input layer and hidden layer
$W \in \mathscr{R}^{N \times H}$ are weights between hidden layer and output layer

(a) On the diagram provided, label each of the pieces of notation defined above in their appropriate places. Focus on understanding the role of each variable in defining your neural net; don't worry too much about the exact details of how to annotate the diagram.

Now, draw in a bias term to both the input layer and the hidden layer and connect it appropriately with the other nodes.

**Solution:** The filled-in neural net with bias terms looks like this:




Here, $x_i$ represents the $i^{th}$ input neuron, also known as the $i^{th}$ feature of our input vector. $B_i$ is the bias neuron for the $i^{th}$ layer. $V_{ij}$ is the weight connecting the $i^{th}$ input neuron with the $j^{th}$ hidden neuron. Remember that $s_i^h$ is the input to the $i^{th}$ hidden neuron, where $s_j^h = \sum_{i=1}^{3} V_{ij} * x_i$. This input is put through the hidden layer's activation function $f(x)$ to get the value of that node. Likewise, $W_{ij}$ is the weight connecting the $i^{th}$ hidden neuron with the $j^{th}$ output neuron, and $s_j^o = \sum_{i=1}^{5} V_{ij} * f(s_i^h)$. Finally, this is passed through the output layer's activation function $g(x)$ to get our final output.

(b) Suppose you want to include a bias term in both the input layer and hidden layer, how many weights will be trained in total now? What will be the dimensions of our weight matrices $V$ and $W$ if we include these bias terms? What if we had $P = p$ input features, $l$ hidden layers each with $H = h$ hidden units, $N = n$ outputs, and bias terms in both the input and hidden layers?

**Solution:** With bias terms:

$$(2+1)\times 4+(4+1)\times 1$$

Our weight matrices will then have the dimensions:

$$V \in \mathscr{R}^{4\times(2+1)} \text{ and } W \in \mathscr{R}^{1\times(4+1)}$$

since $V$ contains the weights between the input layer (with bias) and hidden layer, and $W$ contains the weights between the hidden layer (with bias) and output layer.

Generalized neural network with bias terms:

$$(p+1)h+\max(0,l-1)(h+1)h+(h+1)n$$

which follows the pattern of counting weights by taking the number of nodes in the layer before plus one for for the bias "node" and multiplying it by the number of nodes in the next layer (because our neural net is fully connected), doing this for every pair of adjacent layers.

(c) Use the chain rule to expand the following partial derivatives (no need to find the actual derivatives themselves):

$$\frac{\partial L}{\partial W_{j,k}} =$$
$$\frac{\partial L}{\partial V_{i,j}} =$$

**Solution:** Back propagation is just a nice term for calculating derivatives using the chain rule. **The key to using the chain rule in neural networks is to know which variables are related and which variables are independent of each other**. Since $W_{jk}$ is the weight between neuron $h_j$ in the hidden layer and neuron $z_k$ in the output layer, it's important to see $W_{jk}$ and $z_k$ are related while $W_{jk}$ and $z_{k+1}$ are not. By the same reason,

$$\frac{\partial L}{\partial W_{j,k}} = \frac{\partial L}{\partial z_k}\frac{\partial z_k}{\partial s_k^o}\frac{\partial s_k^o}{\partial W_{j,k}}$$

In the same way, $V_{ij}$ is the weight between each neuron $x_i$ in the input layer and each neuron $h_j$ in the hidden layer and each $h_j$ affects all the neurons in the output layer. Thus when deriving

$\frac{\partial L}{\partial V_{ij}}$, we should apply the chain rule over all output neurons and over only a single neuron $h_j$ in the hidden layer.

$$\frac{\partial L}{\partial V_{i,j}} = \sum_{k=1}^{N} \frac{\partial L}{\partial z_k} \cdot \frac{\partial z_k}{\partial s_k^o} \cdot \frac{\partial s_k^o}{\partial h_j} \cdot \frac{\partial h_j}{\partial s_j^h} \cdot \frac{\partial s_j^h}{\partial V_{i,j}}$$

(d) The softmax function, or normalized exponential function, is a generalization of the logistic function to handle multiclass classification. The softmax function "squashes" a N-dimensional vector $s$ of arbitrary real values to a K-dimensional vector $\sigma(s)$ of real values in the range [0, 1] that add up to 1. The $i$-th value corresponds to the probability that the output is class $i$. The function is given by the following:

$$\sigma(s_j) = \frac{e^{s_j}}{\sum_{k=1}^{N} e^{s_k}} \qquad \text{for } j = 1 \ldots N$$

The cross entropy error is commonly paired with softmax activation in the output layer and is defined as:

$$L(z) = \sum_{k=1}^{N} y_k \ln(z_k)$$

Now, let $g(x)$ be the softmax function and let our loss function be the cross entropy loss. Calculate the partial derivative of $L$ with respect to $W_{i,j}$. (Note: For part (c), you were not required to calculate the actual derivative. For this part, you must do so.)

**Solution:** Notice, that our solution from part (c) is no longer accurate because we had assumed $z_k$ did not depend on $W_{i,j}$ if $k \neq j$. However, by the mechanics of the softmax function, that is no longer true, so we have to sum over all $z_k$.

$$\frac{\partial L}{\partial W_{i,j}} = \sum_{k=1}^{N} \frac{\partial L}{\partial z_k} \frac{\partial z_k}{\partial s_j} \frac{\partial s_j}{\partial W_{i,j}}$$

$$\frac{\partial L}{\partial z_k} = -\frac{y_k}{z_k}$$

$$\frac{\partial z_k}{\partial s_j} = \begin{cases} z_k(1 - z_k) & \text{if } i = k \\ -z_i z_k & \text{if } i \neq k \end{cases}$$

$$\frac{\partial s_j}{\partial W_{i,j}} = h_i$$

$$
\begin{aligned}
\frac{\partial L}{\partial W_{i,j}} &= \sum_{k=1}^{N} \frac{\partial L}{\partial z_k}\frac{\partial z_k}{\partial s_j}\frac{\partial s_j}{\partial W_{i,j}} \\
&= \frac{\partial s_j}{\partial W_{i,j}}\left(\frac{\partial L}{\partial z_j}\frac{\partial z_j}{\partial s_j} + \sum_{k\neq j}^{N} \frac{\partial L}{\partial z_k}\frac{\partial z_k}{\partial s_j}\right) \\
&= \frac{\partial s_j}{\partial W_{i,j}}\left(-\frac{y_j}{z_j} z_j(1-z_j) + \sum_{k\neq j}^{N} -\frac{y_k}{z_k}(-z_j z_k)\right) \\
&= \frac{\partial s_j}{\partial W_{i,j}}\left(-y_j(1-z_j) + \sum_{k\neq j}^{N} y_k z_j\right) \\
&= \frac{\partial s_j}{\partial W_{i,j}}\left(-y_j + y_j z_j + \sum_{k\neq j}^{N} y_k z_j\right) \\
&= \frac{\partial s_j}{\partial W_{i,j}}\left(-y_j + \sum_{k=1}^{N} y_k z_j\right) \\
&= \frac{\partial s_j}{\partial W_{i,j}}\left(-y_j + z_j \sum_{k=1}^{N} y_k\right) \\
&= \frac{\partial s_j}{\partial W_{i,j}}(-y_j + z_j) \\
&= h_i(-y_j + z_j)
\end{aligned}
$$

# 3 Fisher's Linear Discriminant Analysis

*(Inspired by section 4.3.3 of ESL.)*

Suppose we have a binary classification problem for which we would like to find a linear estimator for–that is, given a random vector $\vec{X}$, we would like to find predictor $Z = w^\top \vec{X}$, s.t. we maximize $(E[Z|Y=1] - E[Z|Y=0])^2$.

We define:
$\mu_i = E[X|Y=i]$
$\Sigma = (\vec{X} - \mu_i)(\vec{X} - \mu_i)^\top$, where for both classes $i = 0, 1$ we have the same covariance matrix.
$n_0 = n_1$, i.e. the number of data points in each class are the same.

Thus we have

$$\begin{aligned}
\max(E[Z|Y=1] - E[Z|Y=0])^2 &= \max(E[w^\top \vec{X}|Y=1] - E[w^\top \vec{X}|Y=0])^2 \\
&= \max(w^\top (E[\vec{X}|Y=1] - E[\vec{X}|Y=0]))^2 \\
&= \max(w^\top (\mu_1 - \mu_0))^2 \\
&= \max w^\top (\mu_1 - \mu_0)(\mu_1 - \mu_0)^\top w
\end{aligned}$$

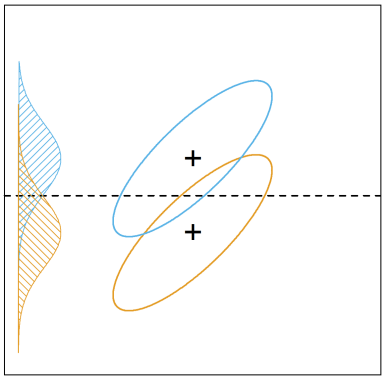
(a) Given the above maximization problem, we know that $|w|$ can increase without bound unless we have some kind of normalization. One way we can normalize is to divide by the magnitude of $w$. What is the result of this normalization?

**Solution:** Normalizing as instructed, we have:

$$\max \frac{w^\top (\mu_1 - \mu_0)(\mu_1 - \mu_0)^\top w}{w^\top w}$$

Note that while this looks similar to the Rayleigh quotient, this is not PCA, since $(\mu_1 - \mu_0)(\mu_1 - \mu_0)^\top$ has a rank of 1.

(b) Is the approach from part (a) equivalent to LDA? Refer to this plot from ESL to explain your conceptual answer. The crosses represent centroids. The ellipses represent an isocontour of the multivariate gaussian corresponding to the covariance matrix of a set of data. The dotted line represents the decision boundary. The bell curves represent a projection of the multivariate gaussians.



**Solution:** No. This approach does not take into account the covariance matrices of the centroids whatsoever. From the plot, we can see that under LDA, the decision boundary should not be the line that is simply equidistant to the two centroids.

(c) Now, we will try normalizing instead by $\frac{1}{2}\text{var}[Z|Y=1]+\frac{1}{2}\text{var}[Z|Y=0]$. Simplify in terms of $\Sigma$ and $\mu_i$ (refer to the definitions at the beginning of this problem).

**Solution:**

$$\begin{aligned}\frac{1}{2}\text{var}[Z|Y=1]+\frac{1}{2}\text{var}[Z|Y=0] &= \text{var}[w^\top \vec{X}|Y] \\ &= E[w^\top(\vec{X}-\mu_i)|Y]^2 \\ &= w^\top(\vec{X}-\mu_i)(\vec{X}-\mu_i)^\top w \\ &= w^\top \Sigma w\end{aligned}$$

Thus we have,

$$\max \frac{w^\top(\mu_1-\mu_0)(\mu_1-\mu_0)^\top w}{w^\top \Sigma w}$$

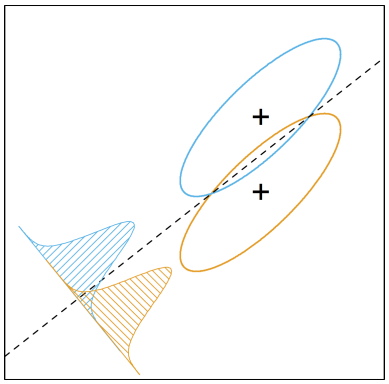
(d) Prove that your approach from part (c) is indeed equivalent to the decision boundary of LDA you are familiar with by finding $w$ s.t. $w^\top x = c$, for some constant $c$. (Take it that solving for $w$ through Lagrange multipliers yields that $w \propto \Sigma^{-1}(\mu_1-\mu_0)$.)
$(x-\mu_0)^\top\Sigma(x-\mu_0) = (x-\mu_1)^\top\Sigma(x-\mu_1) = \ldots$

**Solution:**

$$\begin{aligned}(x-\mu_0)^\top\Sigma^{-1}(x-\mu_0) &= (x-\mu_1)^\top\Sigma^{-1}(x-\mu_1) \\ x^\top\Sigma^{-1}x-\mu_0^\top\Sigma^{-1}x-x^\top\Sigma^{-1}\mu_0+\mu_0^\top\Sigma^{-1}\mu_0 &= x^\top\Sigma^{-1}x-\mu_1^\top\Sigma^{-1}x-x^\top\Sigma^{-1}\mu_1+\mu_1^\top\Sigma^{-1}\mu_1 \\ -2\mu_0^\top\Sigma^{-1}x+\mu_0^\top\Sigma^{-1}\mu_0 &= -2\mu_1^\top\Sigma^{-1}x+\mu_1^\top\Sigma^{-1}\mu_1 \\ 2\mu_1^\top\Sigma^{-1}x-2\mu_0^\top\Sigma^{-1}x &= c \\ \mu_1^\top\Sigma^{-1}x-\mu_0^\top\Sigma^{-1}x &= c \\ (\mu_1-\mu_0)^\top\Sigma^{-1} &\propto w^\top \\ w &\propto \Sigma^{-1}(\mu_1-\mu_0)\end{aligned}$$

Since we found that $w$ is equivalent to what Lagrange multipliers yields, as given to us, we can conclude that these formulations are equivalent.

(e) Interpret why the approach from parts (c) and (d) are conceptually equivalent to LDA? Refer to this plot from ESL to explain your answer.



**Solution:** Yes. $w$ is the axis along which these projected distributions have minimal intersection. This is equivalent to maximizing the variance of the projection of the centroids (i.e. the means of the projected distributions), divided by the projected variance of the distributions themselves.

# 4 Logistic Regression

Consider the log-likelihood function for logistic regression

$$\ell(\theta) = \sum_{i=1}^{m} y^{(i)} \ln h(x^{(i)}) + (1 - y^{(i)}) \ln(1 - h(x^{(i)}))$$

Find the Hessian $H$ of this function , and show that for every $z$, it holds true that

$$z^\top H z \leq 0$$

(Hint: You might want to start by showing that $\sum_i \sum_j z_i x_i x_j z_j = (x^\top z)^2$)

**Solution:** Recall that we have $g'(z) = g(z)(1 - g(z))$, and thus for $h(x) = g(\theta^\top x)$. We have $\frac{\partial h(x)}{\partial \theta_k} = h(x)(1 - h(x))x_k$.

$$\begin{aligned}
\frac{\partial l(\theta)}{\partial \theta_k} &= \sum_{i=1}^{m} (y^{(i)} - h(x^{(i)}))x_k^{(i)} \\
H_{kl} &= \frac{\partial^2 l(\theta)}{\partial \theta_k \partial \theta_l} \\
&= \sum_{i=1}^{m} -\frac{\partial h(x^{(i)})}{\partial \theta_l} x_k^{(i)} \\
&= \sum_{i=1}^{m} -h(x^{(i)})(1 - h(x^{(i)}))x_l^{(i)} x_k^{(i)}
\end{aligned}$$

So we have for the hessian matrix $H$ (using that for $X = xx^\top$ if and only if $X_{ij} = x_i x_j$):

$$H = -\sum_{i=1}^{m} h(x^{(i)})(1 - h(x^{(i)}))x^{(i)}x^{(i)\top}$$

And to prove that $H$ is negative semi-definite, we show $z^\top Hz \leq 0$ for all $z$

$$\begin{aligned} z^\top Hz &= -z^\top (\sum_{i=1}^{m} h(x^{(i)})(1 - h(x^{(i)}))x^{(i)}x^{(i)\top})z \\ &= -\sum_{i=1}^{m} h(x^{(i)})(1 - h(x^{(i)}))z^\top x^{(i)}x^{(i)\top}z \\ &= -\sum_{i=1}^{m} h(x^{(i)})(1 - h(x^{(i)}))(z^\top x^{(i)})^2 \\ &\leq 0 \end{aligned}$$

with the last inequality holding, since $0 \leq h(x^{(i)}) \leq 1$, which implies $h(x^{(i)}(1 - h(x^{(i)})) \geq 0$, and $(z^\top x^{(i)})^2 \geq 0$