B555 - Machine Learning - Homework 3

Enrique Areyan
April 13, 2015

Problem 1: Consider a logistic regression problem where $\mathcal{X} = \mathbb{R}^k$ and $\mathcal{Y} = \{-1, +1\}$. Derive the weight update rule that maximizes the likelihood.

Solution: By definition, for this logistic regression problem we have:

$$P(Y = 1|\mathbf{x}, \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x})} \Longrightarrow P(Y = -1|\mathbf{x}, \mathbf{w}) = 1 - \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x})}$$

So the labels Y follow a Bernoulli distribution:

$$P(Y = y|\mathbf{x}, \mathbf{w}) = \begin{cases} \left(\frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x})}\right)^{\frac{1}{2}(y+1)} & \text{for } y = 1 \\ \left(1 - \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x})}\right)^{\frac{1}{2}(1-y)} & \text{for } y = -1 \end{cases}$$

We wish to solve the following optimization problem:

$$\mathbf{w}_{ML} = \arg\max_{\mathbf{w}} \{l(\mathbf{w})\}$$

where,

$$l(\mathbf{w}) = \prod_{i=1}^{n} p(y_i|\mathbf{x}_i, \mathbf{w}) = \prod_{i=1}^{n} \left(\frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x}_i)}\right)^{\frac{1}{2}(y_i+1)} \cdot \left(1 - \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x}_i)}\right)^{\frac{1}{2}(1-y_i)}$$

As usual, we will maximize the log-likelihood instead:

$$\begin{aligned} l(\mathbf{w}) = \log(l(\mathbf{w})) &= \log\left\{\prod_{i=1}^{n} \left(\frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x}_i)}\right)^{\frac{1}{2}(y_i+1)} \cdot \left(1 - \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x}_i)}\right)^{\frac{1}{2}(1-y_i)}\right\} \\ &= \sum_{i=1}^{n} \left\{\tfrac{1}{2}(y_i + 1)\log\left(\frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x}_i)}\right) + \tfrac{1}{2}(1 - y_i)\log\left(1 - \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x}_i)}\right)\right\} \\ &= \sum_{i=1}^{n} \tfrac{1}{2}\left\{(y_i + 1)(-\log\left(1 + \exp(-\mathbf{w}^T\mathbf{x}_i)\right) + (1 - y_i)\left[(-\mathbf{w}^T\mathbf{x}_i - \log\left(1 + \exp(-\mathbf{w}^T\mathbf{x}_i)\right)\right]\right\} \\ &= \sum_{i=1}^{n} \tfrac{1}{2}\left\{(y_i - 1)\mathbf{w}^T\mathbf{x}_i + 2\log\left(\frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x}_i)}\right)\right\} \\ &= \sum_{i=1}^{n} \left\{\tfrac{1}{2}(y_i - 1)\mathbf{w}^T\mathbf{x}_i + \log\left(\frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x}_i)}\right)\right\} \end{aligned}$$

Now we are ready to take derivatives:

$$\begin{aligned} \frac{\partial l(\mathbf{w})}{\partial w_j} &= \sum_{i=1}^{n} \left\{\tfrac{1}{2}(y_i - 1)x_{ij} - \left(\frac{[1 + \exp(-w_j x_{ij})] \cdot \exp(-w_j x_{ij}) \cdot (-x_{ij})}{[1 + \exp(-w_j x_{ij})]^2}\right)\right\} \\ &= \sum_{i=1}^{n} x_{ij}\left[\frac{1}{2}(y_i - 1) + \frac{\exp(-w_j x_{ij})}{1 + \exp(-w_j x_{ij})}\right] \\ &= \sum_{i=1}^{n} x_{ij}\left[\frac{1}{2}(y_i - 1) + 1 - 1 + \frac{\exp(-w_j x_{ij})}{1 + \exp(-w_j x_{ij})}\right] \\ &= \sum_{i=1}^{n} x_{ij}\left[\frac{1}{2}(y_i + 1) - \frac{1}{1 + \exp(-w_j x_{ij})}\right] \end{aligned}$$

Following the notation on the lecture notes we get that: $\frac{\partial ll(\mathbf{w})}{\partial w_j} = \mathbf{f}_j^T \left(\frac{1}{2}(\mathbf{y}+1) - \mathbf{p}\right)$, where $\mathbf{f}_j$ is the j-th column (feature) of data matrix $\mathbf{X}$, $\mathbf{y}$ is an n-dimensional column vector of class labels and $\mathbf{p}$ is an n-dimensional column vector of estimated posterior probabilities $p_i = P(Y_i|\mathbf{x}_i, \mathbf{w})$ for $i = 1, \ldots n$. Note that this is the same solution for logistic regression in case $\mathcal{Y} = \{0, 1\}$, with the only difference that we transform vector $\mathbf{y}$ by adding one to it and dividing by two. More compactly, we have that:

$$\nabla ll(\mathbf{w}) = \mathbf{X}^T \left(\frac{1}{2}(\mathbf{y}+1) - \mathbf{p}\right)$$

Likewise, we need to take second partial derivatives:

$$\begin{aligned}
\frac{\partial^2 ll(\mathbf{w})}{\partial w_j \partial w_k} &= \sum_{i=1}^{n} -x_{ij} \left[-\frac{exp(-w_j x_{ij}) \cdot (-x_{ik})}{(1+\exp(-w_j x_{ij}))^2} \right] \\
&= \text{(Following the notes on the book)} \\
&= -\mathbf{f}_j^T \mathbf{P}(\mathbf{I}-\mathbf{P})\mathbf{f}_k
\end{aligned}$$

So note that at this point, the Hessian for this logistic regression is the same as for the original logistic regression:

$$H_{ll(\mathbf{w})} = -\mathbf{X}^T \mathbf{P}(\mathbf{I}-\mathbf{P})\mathbf{X}$$

Finally, the weight update rule that maximizes the likelihood is

$$\boxed{\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \left(\mathbf{X}^T \mathbf{P}^{(t)}(\mathbf{I}-\mathbf{P}^{(t)})\mathbf{X}\right)^{-1} \mathbf{X}^T \left(\frac{1}{2}(\mathbf{y}+1) - \mathbf{p}^{(t)}\right)}$$

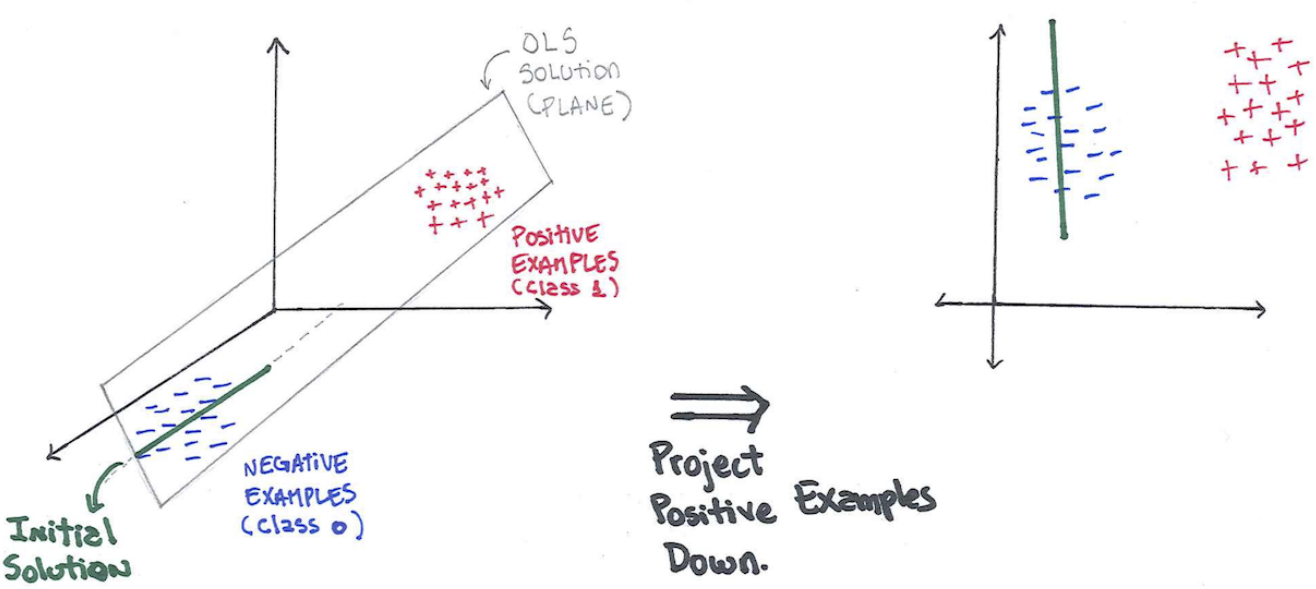
Problem 2: Consider a logistic regression problem with its initial solution obtained through the OLS regression, i.e. $\mathbf{w}^{(0)} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$, in the context of the code provided in class (week 8). Recall that $\mathbf{x}$ had a Gaussian distribution and that $\dim(\mathbf{x})$=2 (before adding a column of ones) and that $y \in \{0, 1\}$. You probably noticed that the initial separation line is consistently closer to the data points of class 0.

a) Why is this the case? Draw a picture (if possible) to support your argument.

b) Devise a better initial solution by modifying the standard formula $\mathbf{w}^{(0)} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$.

c) Now again consider the case where $y \in \{-1, +1\}$. What is the form of the modified solution from part (b) in this case?

Solution: a)



The OLS solution is a perpendicular plane that passes through both positive and negative examples in a 3 dimensional space. However, when we plot the positive and negative examples in a two dimensional space, we project positive examples down to the plane were the negative examples are (recall negative examples have 0 as their label). Hence, the initial separation line we see is the intersection of the OLS solution plane with the plane were the negative examples are. This explains why the initial separation line is consistently closer to the data points of class 0.

b) Instead of using regular OLS for the initial weights, use:

$$\mathbf{w}^{(0)} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{y} - 1/2)$$

i.e., instead of using the label of vector $\mathbf{y}$, subtract $1/2$ to each component of $\mathbf{y}$. This solution works because we are shifting the plane depicted in a) to the right causing the intersection of the plane with the xy plane to be shifted to the right, closer to the positive examples.

c) In this case the form of the solution is that given by the original OLS:

$$\mathbf{w}^{(0)} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Note: in both part b) and c) we are just subtracting the mean of the labels from our vector $\mathbf{y}$ of labels, i.e.,

$$\mathbf{w}^{(0)} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{y} - m)$$

where m is the mean of the labels. For example, if the labels are $\{0, 1\}$, then $m = \dfrac{0+1}{2} = \dfrac{1}{2}$ whereas if the labels are $\{-1, +1\}$, then $m = \dfrac{-1+1}{2} = 0$, exactly recovering our answers for part b) and c).

Problem 3: Consider the same situation as in previous question. We used logistic regression to solve this classification problem, but here we want to compare that solution to the optimal decision surface.

a) Find the optimal decision surface assuming that each class-conditional distribution is defined as a two-dimensional Gauss distribution:

$$p(\mathbf{x}|y = i) = \frac{1}{(2\pi)^{k/2}} \cdot \frac{1}{|\mathbf{\Sigma}_i|^{1/2}} \cdot \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{m}_i)^T \cdot \mathbf{\Sigma}_i^{-1} \cdot (\mathbf{x} - \mathbf{m}_i)\right)$$

where $i \in \{0, 1\}, \mathbf{m}_0 = (1, 2), \mathbf{m}_1 = (6, 3), \mathbf{\Sigma}_0 = \mathbf{\Sigma}_1 = \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix}, P(y = 0) = P(y = 1) = \frac{1}{2}$, and $|\mathbf{\Sigma}_i|$ is the determinant of $\mathbf{\Sigma}_i$.

b) Generalize the solution from part (a) for the case when $\mathbf{m}_0 = (m_{01}, m_{02}), \mathbf{m}_1 = (m_{11}, m_{12})$, $\mathbf{\Sigma}_0 = \mathbf{\Sigma}_1 = \begin{vmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{vmatrix}$, and $P(y = 0) \neq P(y = 1)$.

c) Generalize the solution from part (b) to arbitrary covariance matrices $\mathbf{\Sigma}_0$ and $\mathbf{\Sigma}_1$. Discuss the shape of the optimal decision surface.

Solution: To obtain an optimal decision, let us minimize the risk R for a class i, where $i = 0, 1$ defined by:

$$R(i|\mathbf{x}) = \sum_{j=0}^{1} L(i|y = j) \cdot P(y = j|\mathbf{x})$$

with the zero-one loss function:

$$L(i|y = j) = \begin{cases} 0 & \text{if } i = j \\ 1 & \text{if } i \neq j \end{cases}$$

The risk reduces to:

$$R(0|\mathbf{x}) = L(0|y = 0) \cdot P(y = 0|\mathbf{x}) + L(0|y = 1) \cdot P(y = 1|\mathbf{x}) = P(y = 1|\mathbf{x})$$

$$R(1|\mathbf{x}) = L(1|y = 0) \cdot P(y = 0|\mathbf{x}) + L(1|y = 1) \cdot P(y = 1|\mathbf{x}) = P(y = 0|\mathbf{x})$$

We want to minimize the risk, i.e.,

$$\text{Choose class 0 if and only if: } R(0|\mathbf{x}) < R(1|\mathbf{x}) \iff P(y = 0|\mathbf{x}) > P(y = 1|\mathbf{x})$$

Choose class 1 if and only if: $R(1|\mathbf{x}) < R(0|\mathbf{x}) \iff P(y=1|\mathbf{x}) > P(y=0|\mathbf{x})$

The decision boundary occurs where the inequality becomes equality $P(y=0|\mathbf{x}) = P(y=1|\mathbf{x})$.
Hence, we will solve for this equation. First note that by Bayes rule:

$$P(y=0|\mathbf{x}) = \frac{P(\mathbf{x}|y=0)P(y=0)}{P(\mathbf{x})} \text{ and } P(y=1|\mathbf{x}) = \frac{P(\mathbf{x}|y=1)P(y=1)}{P(\mathbf{x})}$$

From this we get that:

$$\begin{aligned} P(y=0|\mathbf{x}) = P(y=1|\mathbf{x}) \quad &\iff \quad \frac{P(\mathbf{x}|y=0)P(y=0)}{P(\mathbf{x})} = \frac{P(\mathbf{x}|y=1)P(y=1)}{P(\mathbf{x})} \\ &\iff \quad P(\mathbf{x}|y=0)P(y=0) = P(\mathbf{x}|y=1)P(y=1) \qquad \text{canceling constant } P(\mathbf{x}) \end{aligned}$$

Hence, the optimal boundary can be expressed as the solution of the equation:

$$\boxed{P(\mathbf{x}|y=0)P(y=0) = P(\mathbf{x}|y=1)P(y=1)}$$

In each of the following parts a),b) and c), we will solve this equation under different assumptions.

a) Suppose that $P(y=0) = P(y=1) = \frac{1}{2}$. The optimal boundary reduces to $P(\mathbf{x}|y=0) = P(\mathbf{x}|y=1)$.
Let us write each sides of this equation separately. Note that $\mathbf{x} = (x_0, x_1)$:

$$\begin{aligned} P(\mathbf{x}|y=0) &= \frac{1}{2\pi} \cdot \frac{1}{1} \cdot \exp\left(-\frac{1}{2}(\mathbf{x} - [1,2]) \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} (\mathbf{x} - [1,2])\right) \\ &= \frac{1}{2\pi} \exp\left(-\frac{1}{2}([x_0, x_1] - [1,2]) \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} ([x_0, x_1] - [1,2])\right) \\ &= \frac{1}{2\pi} \exp\left(-\frac{1}{2}[x_0 - 1, x_1 - 2] \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} [x_0 - 1, x_1 - 2]\right) \\ &= \frac{1}{2\pi} \exp\left(-\frac{1}{2} \begin{bmatrix} x_0 - 1 \\ x_1 - 2 \end{bmatrix} [x_0 - 1, x_1 - 2]\right) \\ &= \frac{1}{2\pi} \exp\left(-\frac{1}{2}\left((x_0 - 1)^2 + (x_1 - 2)^2\right)\right) \\ &= \frac{1}{2\pi} \exp\left(-\frac{1}{2}\left(x_0^2 - 2x_0 + 1 + x_1^2 - 4x_1 + 4\right)\right) \\ &= \frac{1}{2\pi} \exp\left(-\frac{1}{2}\left(x_0^2 + x_1^2 - 2x_0 - 4x_1 + 5\right)\right) \end{aligned}$$

$$\begin{aligned} P(\mathbf{x}|y=1) &= \frac{1}{2\pi} \cdot \frac{1}{1} \cdot \exp\left(-\frac{1}{2}(\mathbf{x} - [6,3]) \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} (\mathbf{x} - [6,3])\right) \\ &= \frac{1}{2\pi} \exp\left(-\frac{1}{2}([x_0, x_1] - [6,3]) \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} ([x_0, x_1] - [6,3])\right) \\ &= \frac{1}{2\pi} \exp\left(-\frac{1}{2}[x_0 - 6, x_1 - 3] \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} [x_0 - 6, x_1 - 3]\right) \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{2\pi}\exp\left(-\frac{1}{2}\begin{bmatrix} x_0-6 \\ x_1-3 \end{bmatrix}[x_0-6, x_1-3]\right) \\
&= \frac{1}{2\pi}\exp\left(-\frac{1}{2}\left((x_0-6)^2+(x_1-3)^2\right)\right) \\
&= \frac{1}{2\pi}\exp\left(-\frac{1}{2}\left(x_0^2-12x_0+36+x_1^2-6x_1+9\right)\right) \\
&= \frac{1}{2\pi}\exp\left(-\frac{1}{2}\left(x_0^2+x_1^2-12x_0-6x_1+45\right)\right)
\end{aligned}
$$

Hence, the optimal surface is given by:

$$\frac{1}{2\pi}\exp\left(-\frac{1}{2}\left(x_0^2+x_1^2-2x_0-4x_1+5\right)\right) = \frac{1}{2\pi}\exp\left(-\frac{1}{2}\left(x_0^2+x_1^2-12x_0-6x_1+45\right)\right)$$

Cancelling, applying log to both sides and rearranging:

$$-2x_0-4x_1+5 = -12x_0-6x_1+45 \iff \boxed{5x_0+x_1-20=0}$$

So, classify a point (x_0, x_1) as coming from class $y=0$ if and only if $5x_0+x_1-20 \geq 0$ and class 1 o/w.

b) Suppose that $\mathbf{m}_0 = (m_{01}, m_{02}), \mathbf{m}_1 = (m_{11}, m_{12}), \mathbf{\Sigma}_0 = \mathbf{\Sigma}_1 = \begin{vmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{vmatrix}$, and $P(y=0) \neq P(y=1)$.

Note that $\mathbf{\Sigma}_0 = \mathbf{\Sigma}_1 = \begin{vmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{vmatrix} \implies \mathbf{\Sigma}_0^{-1} = \mathbf{\Sigma}_1^{-1} = \begin{vmatrix} \frac{1}{\sigma^2} & 0 \\ 0 & \frac{1}{\sigma^2} \end{vmatrix}$

In this case the priors are distinct, so we want to compute $P(\mathbf{x}|y=i)P(y=i)$ for $i=0,1$

$$
\begin{aligned}
P(\mathbf{x}|y=i)P(y=i) &= \frac{1}{2\pi}\cdot\frac{1}{\sigma^2}\cdot\exp\left(-\frac{1}{2}(\mathbf{x}-[m_{i1},m_{i2}])\begin{bmatrix} \frac{1}{\sigma^2} & 0 \\ 0 & \frac{1}{\sigma^2} \end{bmatrix}(\mathbf{x}-[m_{i1},m_{i2}])\right)P(y=i) \\
&= \frac{1}{2\pi\sigma^2}\exp\left(-\frac{1}{2}([x_0,x_1]-[m_{i1},m_{i2}])\begin{bmatrix} \frac{1}{\sigma^2} & 0 \\ 0 & \frac{1}{\sigma^2} \end{bmatrix}([x_0,x_1]-[m_{i1},m_{i2}])\right)P(y=i) \\
&= \frac{1}{2\pi\sigma^2}\exp\left(-\frac{1}{2}[x_0-m_{i1},x_1-m_{i2}]\begin{bmatrix} \frac{1}{\sigma^2} & 0 \\ 0 & \frac{1}{\sigma^2} \end{bmatrix}[x_0-m_{i1},x_1-m_{i2}]\right)P(y=i) \\
&= \frac{1}{2\pi\sigma^2}\exp\left(-\frac{1}{2}\begin{bmatrix} \frac{1}{\sigma^2}(x_0-m_{i1}) \\ \frac{1}{\sigma^2}(x_1-m_{i2}) \end{bmatrix}[x_0-m_{i1},x_1-m_{i2}]\right)P(y=i) \\
&= \frac{1}{2\pi\sigma^2}\exp\left(-\frac{1}{2\sigma^2}\left[(x_0-m_{i1})^2+(x_1-m_{i2})^2\right]\right)P(y=i)
\end{aligned}
$$

The optimal surface is given by the solution of $P(\mathbf{x}|y=0)P(y=0) = P(\mathbf{x}|y=1)P(y=1)$:

$$\frac{1}{2\pi\sigma^2}\exp\left(-\frac{1}{2\sigma^2}\left[(x_0-m_{01})^2+(x_1-m_{02})^2\right]\right)P(y=0) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{1}{2\sigma^2}\left[(x_0-m_{11})^2+(x_1-m_{12})^2\right]\right)P(y=1)$$

Cancelling $\frac{1}{2\pi\sigma^2}$:

$$\exp\left(-\frac{1}{2\sigma^2}\left[(x_0-m_{01})^2+(x_1-m_{02})^2\right]\right)P(y=0) = \exp\left(-\frac{1}{2\sigma^2}\left[(x_0-m_{11})^2+(x_1-m_{12})^2\right]\right)P(y=1)$$

Apply log to both sides and rearrange:

$$-\frac{1}{2\sigma^2}\left[(x_0-m_{01})^2+(x_1-m_{02})^2\right]+\log(P(y=0))=-\frac{1}{2\sigma^2}\left[(x_0-m_{11})^2+(x_1-m_{12})^2\right]+\log(P(y=1)) \implies$$

$$-\frac{1}{2\sigma^2}\left\{\left[(x_0-m_{01})^2+(x_1-m_{02})^2\right]-\left[(x_0-m_{11})^2+(x_1-m_{12})^2\right]\right\}+\log(P(y=0))-\log(P(y=1))=0 \implies$$

$$-\frac{1}{2\sigma^2}\left\{\left[(x_0-m_{01})^2+(x_1-m_{02})^2\right]-\left[(x_0-m_{11})^2+(x_1-m_{12})^2\right]\right\}+\log\left(\frac{P(y=0)}{P(y=1)}\right)=0 \implies$$

$$-\frac{1}{2\sigma^2}\left[x_0^2-2x_0m_{01}+m_{01}^2+x_1^2-2x_1m_{02}+m_{02}^2-x_0^2+2x_0m_{11}-m_{11}^2-x_1^2+2x_1m_{12}-m_{12}^2\right]+\log\left(\frac{P(y=0)}{P(y=1)}\right)=0 \implies$$

$$-\frac{1}{2\sigma^2}\left[-2x_0m_{01}+m_{01}^2-2x_1m_{02}+m_{02}^2+2x_0m_{11}-m_{11}^2+2x_1m_{12}-m_{12}^2\right]+\log\left(\frac{P(y=0)}{P(y=1)}\right)=0$$

Finally, rearranging the equation we obtain the decision boundary:

$$\boxed{\left(\frac{m_{01}-m_{11}}{\sigma^2}\right)x_0+\left(\frac{m_{02}-m_{12}}{\sigma^2}\right)x_1+\frac{(m_{11}^2+m_{12}^2)-(m_{01}^2+m_{02}^2)}{2\sigma^2}+\log\left(\frac{P(y=0)}{P(y=1)}\right)=0}$$

Note that this solution is still a hyperplane. Once can check that this solution works for part (a).

c) In general, the solution is given by the solution of

$$p(\mathbf{x}|y=0)P(y=0)=p(\mathbf{x}|y=1)P(y=1) \implies$$

$$\frac{1}{(2\pi)}\cdot\frac{1}{|\mathbf{\Sigma}_0|^{1/2}}\cdot\exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m}_0)^T\cdot\mathbf{\Sigma}_0^{-1}\cdot(\mathbf{x}-\mathbf{m}_0)\right)P(y=0)=\frac{1}{(2\pi)}\cdot\frac{1}{|\mathbf{\Sigma}_1|^{1/2}}\cdot\exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m}_1)^T\cdot\mathbf{\Sigma}_1^{-1}\cdot(\mathbf{x}-\mathbf{m}_1)\right)P(y=1)$$

Cancelling $\frac{1}{2\pi}$:

$$\frac{1}{|\mathbf{\Sigma}_0|^{1/2}}\cdot\exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m}_0)^T\cdot\mathbf{\Sigma}_0^{-1}\cdot(\mathbf{x}-\mathbf{m}_0)\right)P(y=0)=\frac{1}{|\mathbf{\Sigma}_1|^{1/2}}\cdot\exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m}_1)^T\cdot\mathbf{\Sigma}_1^{-1}\cdot(\mathbf{x}-\mathbf{m}_1)\right)P(y=1)$$

Apply log to both sides:

$$\log(\frac{1}{|\mathbf{\Sigma}_0|^{1/2}})+\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m}_0)^T\cdot\mathbf{\Sigma}_0^{-1}\cdot(\mathbf{x}-\mathbf{m}_0)\right)+\log(P(y=0))=\log(\frac{1}{|\mathbf{\Sigma}_1|^{1/2}})+\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m}_1)^T\cdot\mathbf{\Sigma}_1^{-1}\cdot(\mathbf{x}-\mathbf{m}_1)\right)+\log(P(y=1))$$

Rearranging the equation:

$$\log(\frac{1}{|\mathbf{\Sigma}_0|^{1/2}})+\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m}_0)^T\cdot\mathbf{\Sigma}_0^{-1}\cdot(\mathbf{x}-\mathbf{m}_0)\right)+\log(P(y=0))-\log(\frac{1}{|\mathbf{\Sigma}_1|^{1/2}})-\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m}_1)^T\cdot\mathbf{\Sigma}_1^{-1}\cdot(\mathbf{x}-\mathbf{m}_1)\right)-\log(P(y=1))=0$$

OR

$$\frac{1}{2}\left[(\mathbf{x}-\mathbf{m}_1)^T\mathbf{\Sigma}_1^{-1}(\mathbf{x}-\mathbf{m}_1)-(\mathbf{x}-\mathbf{m}_0)^T\mathbf{\Sigma}_0^{-1}(\mathbf{x}-\mathbf{m}_0)\right]+\frac{1}{2}\left[\log(|\mathbf{\Sigma}_1|)-\log(|\mathbf{\Sigma}_0|)\right]+\log(P(y=0))-\log(P(y=1))=0$$

OR

$$\boxed{\frac{1}{2}\left[(\mathbf{x}-\mathbf{m}_1)^T\mathbf{\Sigma}_1^{-1}(\mathbf{x}-\mathbf{m}_1)-(\mathbf{x}-\mathbf{m}_0)^T\mathbf{\Sigma}_0^{-1}(\mathbf{x}-\mathbf{m}_0)\right]+\frac{1}{2}\left[\log\left(\frac{|\mathbf{\Sigma}_1|}{|\mathbf{\Sigma}_0|}\right)\right]+\log\left(\frac{P(y=0)}{P(y=1)}\right)=0}$$

The last two terms of this equation are constants. The first term is the critical term as it determines the shape of the optimal decision surface. Since the covariances are not the same, we are not guaranteed to cancel quadratic terms. Therefore, the shape, as stated in [3]: "Are hyperquadrics: can be hyperplanes, hyperplane pairs, hyperspheres, hyperellipsoids, hyperparabaloids, hyperhyperparabaloids", depending on the covariances.

Problem 4: We showed in class that the perceptron training rule guarantees convergence of the training algorithm when data sets are linearly separable. Your task in this exercise is to verify this claim experimentally and comment on what you observe. Start by implementing perceptron training, e.g. by using the code from class but modify the data generation code (say, from the EM algorithm code, logistic regression, etc.), to incorporate data of several different dimensionalities $2 \leq k \leq 100$ and data set sizes $100 \leq n \leq 10000$ (for simplicity use that the number of positive and negative examples are roughly identical). Using Gaussian class-conditional distributions will suffice but if you prefer to use other data generators, do not hesitate to do so. Use the logistic regression algorithm to keep only those data sets that are linearly separable (logistic regression may not perform this task perfectly, but it is good enough) and use its solution to obtain the set of "true" coefficients $\mathbf{w}_0$ and parameter ϵ from the class lecture notes #9.

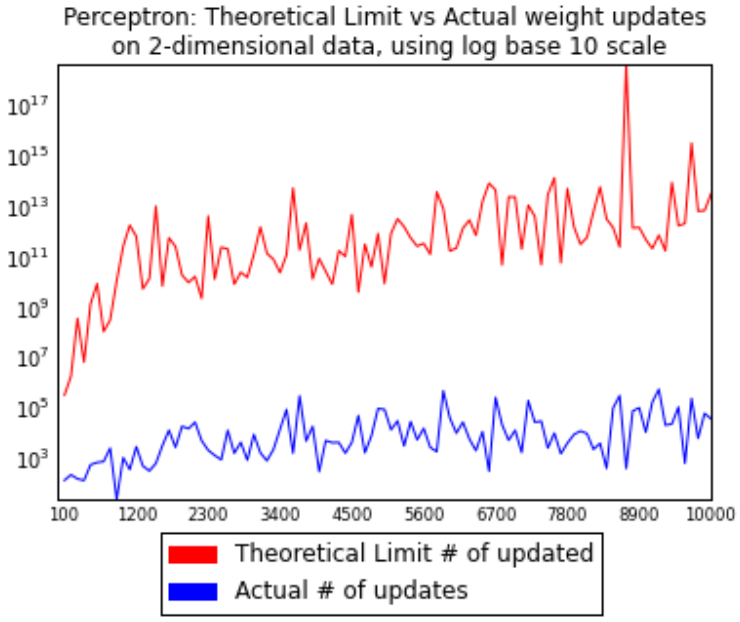
a) Record the number of weight updates (upon misclassification) and compare it with the theoretical limit l_{max} for each data set. Find an appropriate way to summarize (e.g. visualize) your recordings over many runs to support your claims. What do you observe? How tight is the theoretical limit? How does the total number of weight updates change with parameter k? Comment on all your findings.

b) Repeat the process from step (a) for one chosen parameter $2 \leq k \leq 10$. Now explore the dependency of the number of weight updates with the maximum norm M of any data point in your data set. For each data set, construct a "normalized" replica with a controlled norm (smaller or larger) and run the perceptron training algorithm. Does the algorithm behave as expected. Comment on all your findings.

c) Introduce the learning rate parameter $\eta \in (0, 1]$ into the perceptron update rule. What changes are you noticing when η is used. Comment on all your findings.

Solution: **Note:** the code for this problem (and all others) is attached to the OnCourse submission.

a) **Note:** Please take a look at the **Appendix** for more graphs.
The claim is that, in general, the theoretical limit is a very loose. Consider the following graph:

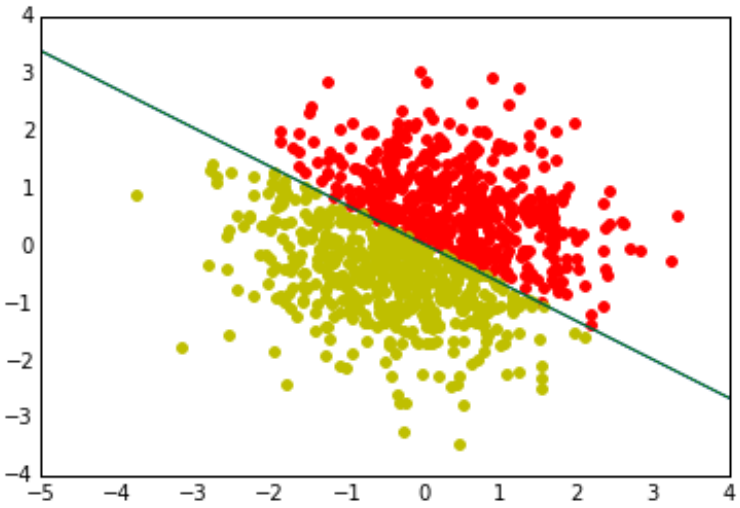


The above graph compares the theoretical limit on the weights updates on a Perceptron against the actual number of updates. Points on the red line correspond to theoretical limit for different data sizes as labeled on the x-axis and points on the blue line correspond to actual weight updates. The idea is that for a given data set we compute the theoretical limit and the actual number of updates obtaining two points, one red and one blue. The line provides a visual aid, points are computed only for data sets of sizes $N = 100, 200, \ldots, 10,000$.

It is readily visible that, for this data, the theoretical limit is very loose and in practice the algorithm uses fewer weight updates to converge. This is even more evident if you consider that the graph is in log scale (base 10) so that the difference between the theoretical limit and the actual number of updates is very high. Note also that the graph shows a linear trend which is consistent with the fact that bigger

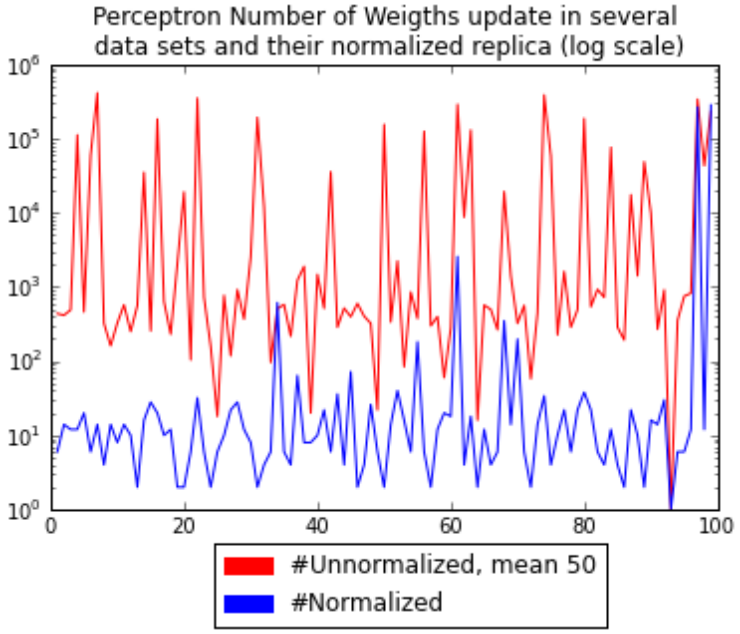
data sets will required more updates both in theory and practice. The appendix provides more evidence to the claim that in general the theoretical limit is a very loose. There you will find very similar graphs for data of dimensions $d = 3, 4, \ldots 10$.

For this test all data sets were constructed from a multivariable normal distribution with mean $\mathbf{1}$ and standard deviation $\mathbf{0}$. In other words, all dimensions are standard normal distribution. A logistic regression is run on the data and points relabeled according to the separating hyperplane provided by the logistic regression. This process generates highly clustered but linearly separable data as the next graph in 2 dimensions and 1,000 data points shows:



In fact, data is so close together that it was not computationally efficient to get results for dimensions bigger than 10 (although it is possible). For the sake of brevity I only present dimensions from 2 to 10, but evidence suggest these results will probably hold for higher dimensions.

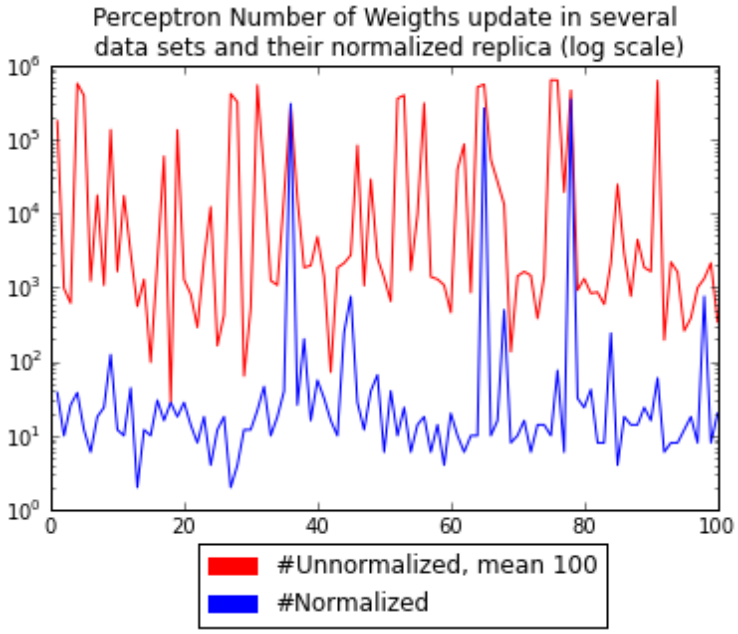
b) I am choosing $k = 2$ (2-dimensional data). The following graph shows 100 different runs on different data sets each of a fixed size $N = 50$, generated from a multi normal distribution with mean 50 on each dimension and the same run on a normalized replica where all points are of norm 1. In general, the maximum norm of the original data sets is around 72. Note that the graph is on a log scale, base 10.



Here we can clearly see that the algorithm behaves as expected. In general for this data set perceptron takes considerably less time on the normalized data set than on the original, unnormalized data set. This

follows from the theory. Consider the theoretical bound: $l_{max} < \frac{M^2}{\epsilon^2}$, in case $M = 1$ we get $l_{max} < \frac{1}{\epsilon^2}$. The unnormalized bound is bigger than the normalized bound by a factor of M^2, which, depending on the maximum norm of the data might make a very big difference. Note that in some instances (in this case very few) the algorithm might make the same number of weights updates (or even more) in the normalized data compared to its unnormalized counterpart. However, this is rarely the case for this data and it is not a contradiction on the theory as we only have an upper bound and not an exact number of weights updates.

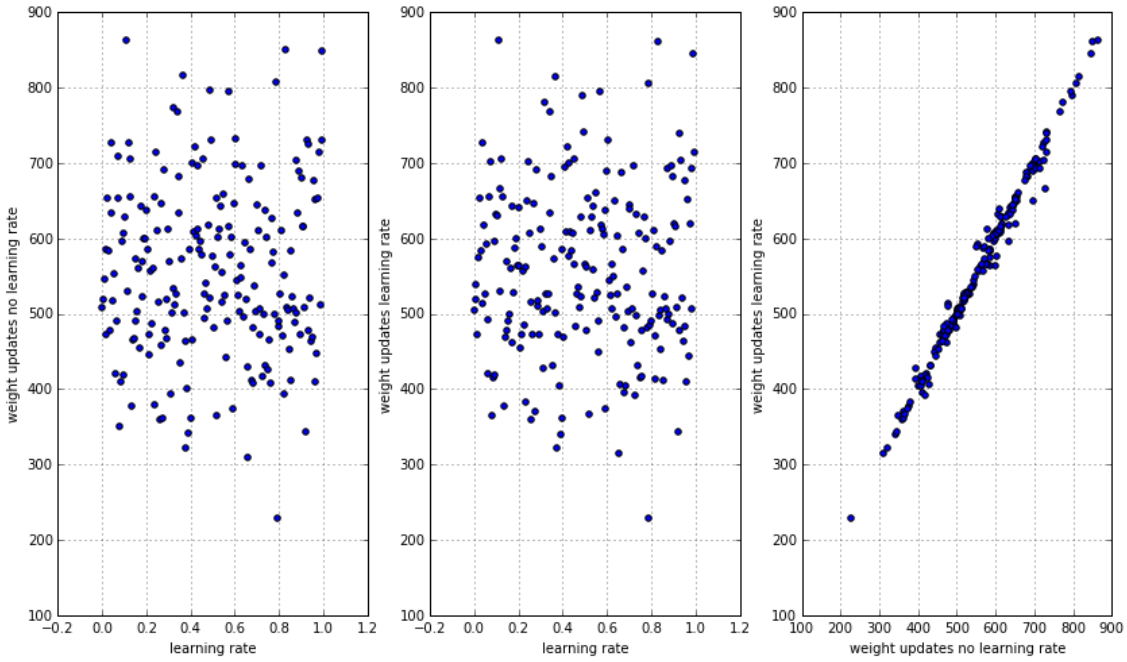
A further example of this behavior can be seen in the next graph which shows again 100 different runs on different data sets each of a fixed size $N = 100$ generated from a multi normal distribution with mean 100 on each dimension and the same run on a normalized replica where all points are of norm 1.



In this case the maximum norm for the unnormalized chart is usually around 144. We can see that the red line here is consistently higher than the blue line in the previous chart. This is expected: the mean of these data is higher and thus, the theoretical bound is higher. Note also that on rare occasions the blue line touches or passes the red line. This is also expected as the data was generated randomly and is perfectly plausible that in certain occasions the unnormalized data set takes fewer weight update than its normalized counterpart. However, as already mentioned, in general we expect (and observe) the opposite behavior.

c) Let us now introduce learning rate $\eta \in (0, 1]$. For simplicity I will analyze work with a fixed, 2 dimensional data set of 500 points. My experiments suggest that the learning rate makes a slight difference on the number of weight updates upon misclassification. However, the effect does not really seem to be that important. I believe this is due to the fact that the data was generated in a way that the effect of the learning rate is minimal.

The following two left-most scatter plots shows points (η, u) where $\eta \in (0, 1]$ is the learning rate and u is the total number of updates upon misclassification of any data point on a data set. The first plot has is in the case where the learning rate is fixed $\eta = 1$, or in other words we use no learning rate. The second plot shows different values for η going from 0.005 to 1 in increments on 0.005. The third plot shows the number of updates upon misclassification when we use no learning rate ($\eta = 1$), and when we vary η.



Problem 5: Consider a classification problem where $\mathcal{X} = \mathbb{R}^k$ and $\mathcal{Y} = \{0, 1\}$. Based on your understanding of the maximum likelihood estimation of weights in logistic regression, develop a linear classifier that models the posterior probability of the positive class as

$$P(y = 1|\mathbf{x}_i, \mathbf{w}) = \frac{1}{2}\left(1 + \frac{\mathbf{w}^T\mathbf{x}_i}{\sqrt{1 + (\mathbf{w}^T\mathbf{x}_i)^2}}\right)$$

Implement the iterative weight update rule and compare the final decision line between logistic regression and this new linear classifier (use at least 10 different data sets to draw your conclusions).

Solution: Our Bernoulli probabilities in this case are (since they have to add up to 1):

$$P(y = 1|\mathbf{x}_i, \mathbf{w}) = \frac{1}{2}\left(1 + \frac{\mathbf{w}^T\mathbf{x}_i}{\sqrt{1 + (\mathbf{w}^T\mathbf{x}_i)^2}}\right) \Longrightarrow P(y = 0|\mathbf{x}_i, \mathbf{w}) = 1 - \frac{1}{2}\left(1 + \frac{\mathbf{w}^T\mathbf{x}_i}{\sqrt{1 + (\mathbf{w}^T\mathbf{x}_i)^2}}\right)$$

Let us construct the likelihood function assuming data points are independent:

$$\begin{aligned} l(\mathbf{w}) &= \prod_{i=1}^{n} P(y_i = 1|\mathbf{x}_i, \mathbf{w})^{y_i} \cdot P(y_i = 0|\mathbf{x}_i, \mathbf{w})^{1-y_i} \\ &= \prod_{i=1}^{n} \left[\frac{1}{2}\left(1 + \frac{\mathbf{w}^T\mathbf{x}_i}{\sqrt{1 + (\mathbf{w}^T\mathbf{x}_i)^2}}\right)\right]^{y_i} \cdot \left[1 - \frac{1}{2}\left(1 + \frac{\mathbf{w}^T\mathbf{x}_i}{\sqrt{1 + (\mathbf{w}^T\mathbf{x}_i)^2}}\right)\right]^{1-y_i} \\ &= \prod_{i=1}^{n} \left[\frac{1}{2}\left(1 + \frac{\mathbf{w}^T\mathbf{x}_i}{\sqrt{1 + (\mathbf{w}^T\mathbf{x}_i)^2}}\right)\right]^{y_i} \cdot \left[\frac{1}{2}\left(1 - \frac{\mathbf{w}^T\mathbf{x}_i}{\sqrt{1 + (\mathbf{w}^T\mathbf{x}_i)^2}}\right)\right]^{1-y_i} \end{aligned}$$

As usual, it is easier to work with the log-likelihood:

$$
\begin{aligned}
ll(\mathbf{w}) &= \log(ll(\mathbf{w})) \\
&= \log\left\{\prod_{i=1}^{n}\left[\frac{1}{2}\left(1+\frac{\mathbf{w}^T\mathbf{x}_i}{\sqrt{1+(\mathbf{w}^T\mathbf{x}_i)^2}}\right)\right]^{y_i}\cdot\left[\frac{1}{2}\left(1-\frac{\mathbf{w}^T\mathbf{x}_i}{\sqrt{1+(\mathbf{w}^T\mathbf{x}_i)^2}}\right)\right]^{1-y_i}\right\} \\
&= \sum_{i=1}^{n}\log\left\{\left[\frac{1}{2}\left(1+\frac{\mathbf{w}^T\mathbf{x}_i}{\sqrt{1+(\mathbf{w}^T\mathbf{x}_i)^2}}\right)\right]^{y_i}\right\}+\log\left\{\left[\frac{1}{2}\left(1-\frac{\mathbf{w}^T\mathbf{x}_i}{\sqrt{1+(\mathbf{w}^T\mathbf{x}_i)^2}}\right)\right]^{1-y_i}\right\} \\
&= \sum_{i=1}^{n} y_i\log(1/2)+y_i\log\left(1+\frac{\mathbf{w}^T\mathbf{x}_i}{\sqrt{1+(\mathbf{w}^T\mathbf{x}_i)^2}}\right)+(1-y_i)\log(1/2)+(1-y_i)\log\left(1-\frac{\mathbf{w}^T\mathbf{x}_i}{\sqrt{1+(\mathbf{w}^T\mathbf{x}_i)^2}}\right) \\
&= n\log(1/2)+\sum_{i=1}^{n} y_i\log\left(1+\frac{\mathbf{w}^T\mathbf{x}_i}{\sqrt{1+(\mathbf{w}^T\mathbf{x}_i)^2}}\right)+(1-y_i)\log\left(1-\frac{\mathbf{w}^T\mathbf{x}_i}{\sqrt{1+(\mathbf{w}^T\mathbf{x}_i)^2}}\right)
\end{aligned}
$$

Now we can optimize this equation to find solutions for the weights $\mathbf{w}$:

$$
\begin{aligned}
\frac{\partial ll(\mathbf{w})}{\partial w_j} &= \sum_{i=1}^{n} y_i\frac{x_{ij}(\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}-\mathbf{w}^T\mathbf{x}_i)}{(\mathbf{w}^T\mathbf{x}_i)^2+1}-(1-y_i)\frac{x_{ij}(\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}+\mathbf{w}^T\mathbf{x}_i)}{(\mathbf{w}^T\mathbf{x}_i)^2+1} \\
&= \sum_{i=1}^{n}\frac{y_i x_{ij}(\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}-\mathbf{w}^T\mathbf{x}_i)-(1-y_i)x_{ij}(\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}+\mathbf{w}^T\mathbf{x}_i)}{(\mathbf{w}^T\mathbf{x}_i)^2+1} \\
&= \sum_{i=1}^{n}\frac{y_i x_{ij}\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}-\cancel{y_i x_{ij}\mathbf{w}^T\mathbf{x}_i}-x_{ij}\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}-x_{ij}\mathbf{w}^T\mathbf{x}_i+y_i x_{ij}\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}+\cancel{y_i x_{ij}\mathbf{w}^T\mathbf{x}_i}}{(\mathbf{w}^T\mathbf{x}_i)^2+1} \\
&= \sum_{i=1}^{n}\frac{y_i x_{ij}\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}-x_{ij}\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}-x_{ij}\mathbf{w}^T\mathbf{x}_i+y_i x_{ij}\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}}{(\mathbf{w}^T\mathbf{x}_i)^2+1} \\
&= \sum_{i=1}^{n}\frac{2y_i x_{ij}\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}-x_{ij}\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}-x_{ij}\mathbf{w}^T\mathbf{x}_i}{(\mathbf{w}^T\mathbf{x}_i)^2+1} \\
&= \sum_{i=1}^{n}\frac{(2y_i-1)x_{ij}\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}-x_{ij}\mathbf{w}^T\mathbf{x}_i}{(\mathbf{w}^T\mathbf{x}_i)^2+1} \\
&= \sum_{i=1}^{n}\frac{x_{ij}\left[(2y_i-1)\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}-\mathbf{w}^T\mathbf{x}_i\right]}{(\mathbf{w}^T\mathbf{x}_i)^2+1}
\end{aligned}
$$

Obviously we won't be able to find an analytical solution for this equation being equal to zero.
Thus, let us proceed as usual, i.e., find the hessian or second partial:

$$
\begin{aligned}
&\frac{\partial^2 ll(\mathbf{w})}{\partial w_j \partial w_k} \\
= \quad & \frac{\partial}{\partial w_k} \sum_{i=1}^{n} \frac{x_{ij}\left[(2y_i-1)\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}-\mathbf{w}^T\mathbf{x}_i\right]}{(\mathbf{w}^T\mathbf{x}_i)^2+1} \\
= \quad & \sum_{i=1}^{n} \frac{\partial}{\partial w_k}\left\{\frac{x_{ij}\left[(2y_i-1)\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}-\mathbf{w}^T\mathbf{x}_i\right]}{(\mathbf{w}^T\mathbf{x}_i)^2+1}\right\} \\
= \quad & \sum_{i=1}^{n} \frac{\frac{\partial}{\partial w_k}\left\{x_{ij}\left[(2y_i-1)\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}-\mathbf{w}^T\mathbf{x}_i\right]\right\}\left[(\mathbf{w}^T\mathbf{x}_i)^2+1\right]-\left\{x_{ij}\left[(2y_i-1)\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}-\mathbf{w}^T\mathbf{x}_i\right]\right\}\frac{\partial}{\partial w_k}\left[(\mathbf{w}^T\mathbf{x}_i)^2+1\right]}{((\mathbf{w}^T\mathbf{x}_i)^2+1)^2} \\
= \quad & \sum_{i=1}^{n} \frac{\frac{\partial}{\partial w_k}\left\{x_{ij}\left[(2y_i-1)\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}-\mathbf{w}^T\mathbf{x}_i\right]\right\}\left[(\mathbf{w}^T\mathbf{x}_i)^2+1\right]}{((\mathbf{w}^T\mathbf{x}_i)^2+1)^2} - \frac{\left\{x_{ij}\left[(2y_i-1)\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}-\mathbf{w}^T\mathbf{x}_i\right]\right\}\frac{\partial}{\partial w_k}\left[(\mathbf{w}^T\mathbf{x}_i)^2+1\right]}{((\mathbf{w}^T\mathbf{x}_i)^2+1)^2}
\end{aligned}
$$

For a fixed i, let us write each of the two above fractions separately:

For the first fraction, first work the partial derivative on the numerator:

$$
\begin{aligned}
\frac{\partial}{\partial w_k}\left\{x_{ij}\left[(2y_i-1)\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}-\mathbf{w}^T\mathbf{x}_i\right]\right\} &= x_{ij}\left\{(2y_i-1)\frac{\partial}{\partial w_k}\left[\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}\right]-\frac{\partial}{\partial w_k}\left[\mathbf{w}^T\mathbf{x}_i\right]\right\} \\
&= x_{ij}\left\{(2y_i-1)\frac{1}{2}\frac{2\mathbf{w}^T\mathbf{x}_i\cdot x_{ik}}{\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}}-x_{ik}\right\} \\
&= x_{ij}x_{ik}\left\{\frac{(2y_i-1)\mathbf{w}^T\mathbf{x}_i}{\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}}-1\right\} \\
&= x_{ij}x_{ik}\left\{\frac{(2y_i-1)\mathbf{w}^T\mathbf{x}_i-\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}}{\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}}\right\}
\end{aligned}
$$

So, the first fraction can be reduced to:

$$
\begin{aligned}
\frac{\frac{\partial}{\partial w_k}\left\{x_{ij}\left[(2y_i-1)\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}-\mathbf{w}^T\mathbf{x}_i\right]\right\}\left[(\mathbf{w}^T\mathbf{x}_i)^2+1\right]}{((\mathbf{w}^T\mathbf{x}_i)^2+1)^2} &= \frac{x_{ij}x_{ik}\left\{\frac{(2y_i-1)\mathbf{w}^T\mathbf{x}_i-\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}}{\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}}\right\}}{(\mathbf{w}^T\mathbf{x}_i)^2+1} \\
&= x_{ij}x_{ik}\left\{\frac{(2y_i-1)\mathbf{w}^T\mathbf{x}_i-\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}}{((\mathbf{w}^T\mathbf{x}_i)^2+1)^{3/2}}\right\}
\end{aligned}
$$

The second fraction is:

$$
\frac{\left\{x_{ij}\left[(2y_i-1)\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}-\mathbf{w}^T\mathbf{x}_i\right]\right\}\frac{\partial}{\partial w_k}\left[(\mathbf{w}^T\mathbf{x}_i)^2+1\right]}{((\mathbf{w}^T\mathbf{x}_i)^2+1)^2} = \frac{x_{ij}x_{ik}2\mathbf{w}^T\mathbf{x}_i\left[(2y_i-1)\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}-\mathbf{w}^T\mathbf{x}_i\right]}{((\mathbf{w}^T\mathbf{x}_i)^2+1)^2}
$$

Combining the two fractions and factoring $x_{ij}x_{ik}$, we get:

$$
\frac{\partial^2 ll(\mathbf{w})}{\partial w_j \partial w_k} = \sum_{i=1}^{n} x_{ij}x_{ik}\left\{\frac{(2y_i-1)\mathbf{w}^T\mathbf{x}_i-\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}}{((\mathbf{w}^T\mathbf{x}_i)^2+1)^{3/2}} - \frac{2\mathbf{w}^T\mathbf{x}_i\left[(2y_i-1)\sqrt{(\mathbf{w}^T\mathbf{x}_i)^2+1}-\mathbf{w}^T\mathbf{x}_i\right]}{((\mathbf{w}^T\mathbf{x}_i)^2+1)^2}\right\}
$$

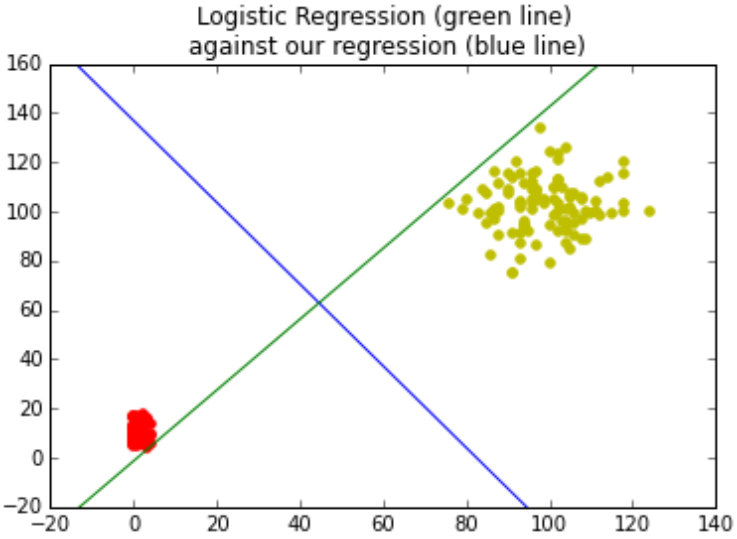
Now we have all the information we need to construct the method:

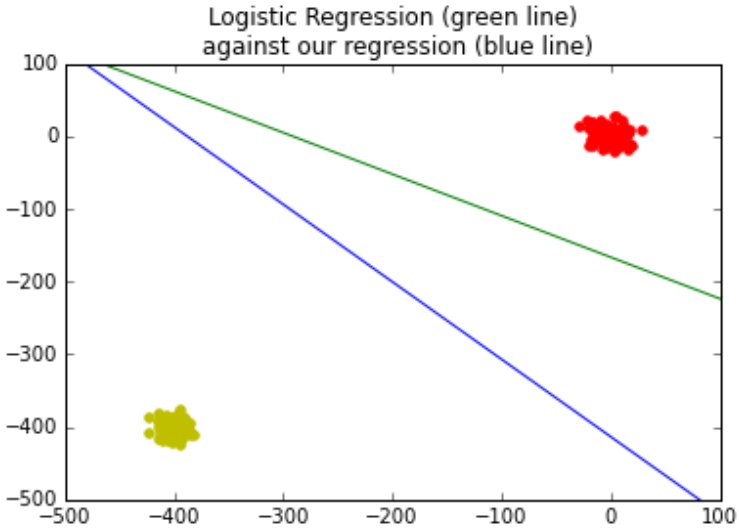
$$\boxed{\mathbf{w}^{(t+1)} = \mathbf{w}^{(0)} - \mathbf{H}^{-1} \cdot \nabla}$$

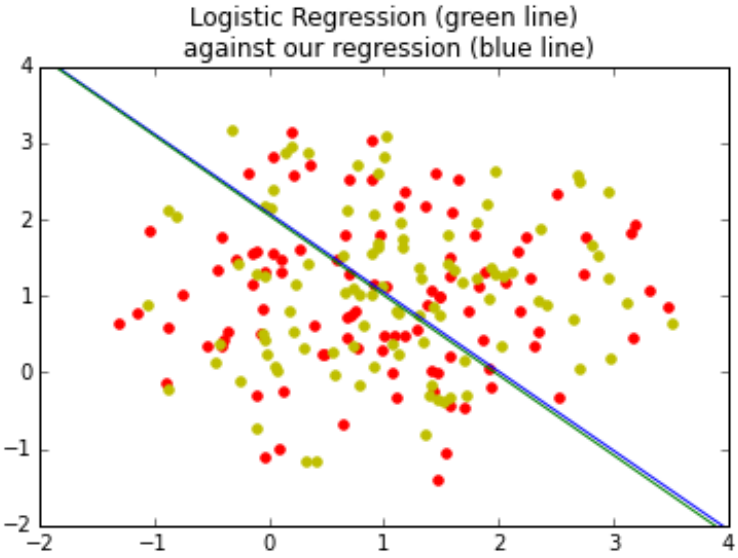
Where the Hessian $\mathbf{H}$ and the gradient ∇, are given by the above equations.

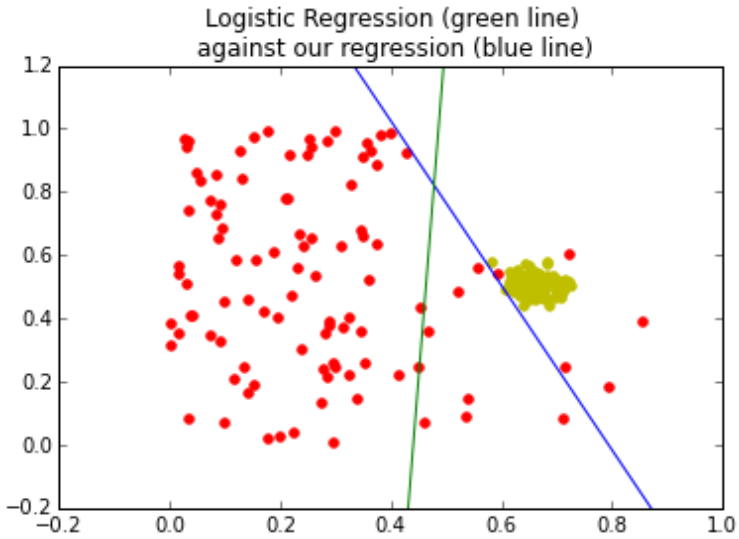
Note: you will find the implementation of this method in the file quadratic_regression.py.

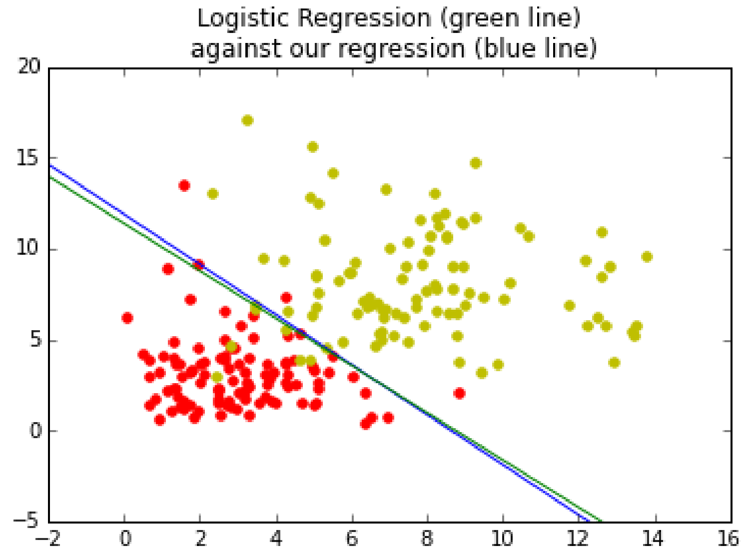
Let us now compare this classificatory with logistic regression for 10 different data sets.

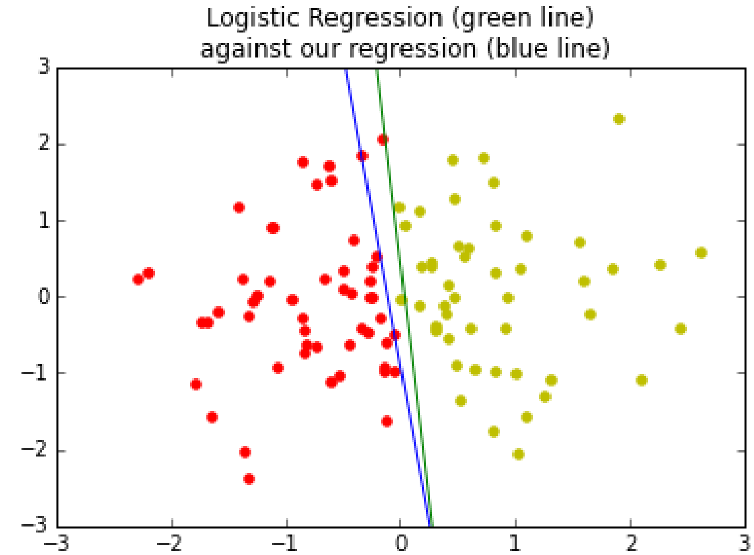


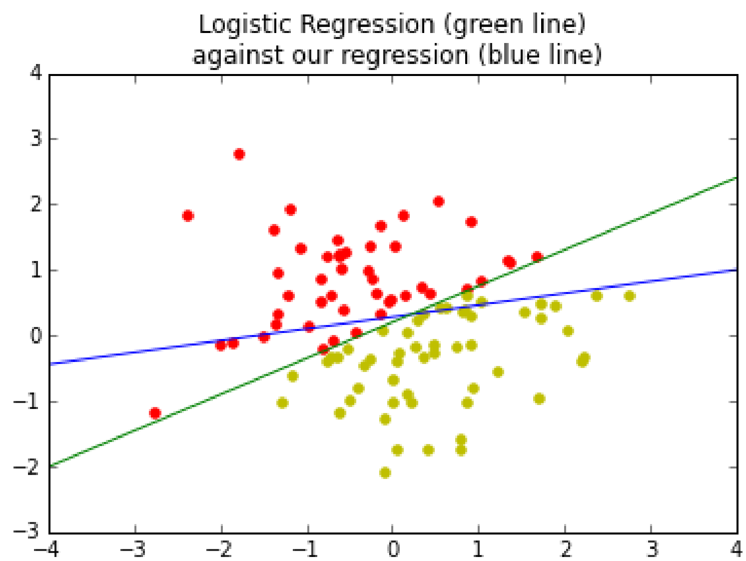


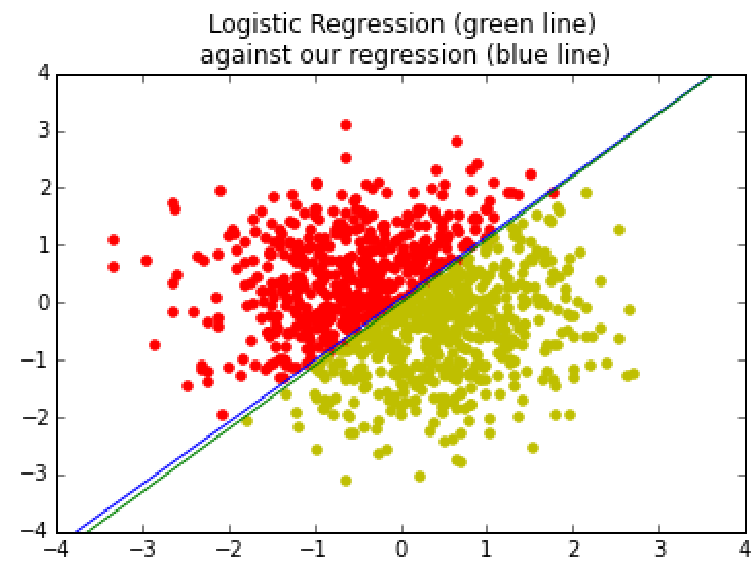


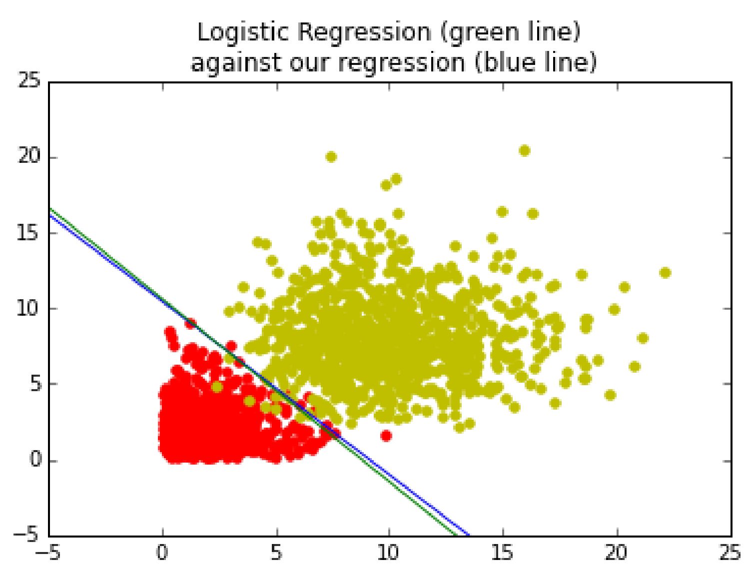


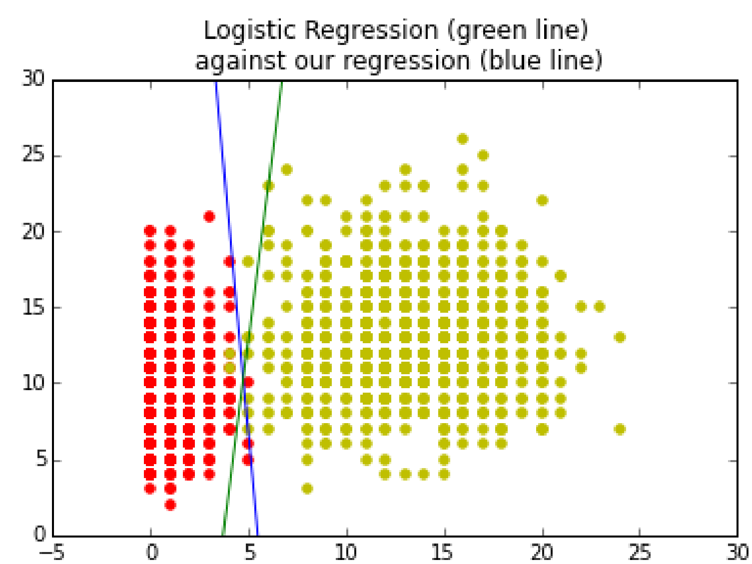












First notice that when we have high clustered, big data sets, the two solutions seems to be approximately the same. Another way of saying this is that one solution converges to the other. However, when data sets are linearly separable with each cluster of positive and negative examples separated, the two solutions can diverge quite a bit. This also

happens with data sets with fewer than 100 points. Finally, if the data set is mixed up then the solutions seem to be approximately the same.

This evidence suggests that logistic regression and our classifier are usually the same for data sets in which there is little hope of having a linear separator or for very highly clustered but linearly separable data. The two classificators differ when the positives and negative examples are very far apart. It seems like when there are many different choices for a linear separator these two classifier make different decisions.

Problem 6: Ordinary Least Squares (OLS) linear regression vs. Generalized Linear Models (GLMs).

a) Use data set crab.xlsx to develop a linear regression model for predicting the number of male satellites on a female's back. What are the sum-of-squares and mean-square-error of the OLS fit.

b) Because the target variable (number of male satellites) is a non-negative integer, it may be beneficial to model it by a Poisson distribution, instead of Gaussian that is a base distribution in OLS linear regression. Read the Generalized Linear Models section from class notes (Instructor's Notes #6). Verify the correctness of the weight update rule for a loglinear link function with Poisson distribution and apply it to the crab data set. Compare both residual sum of squares and mean square error for the fits you achieved using OLS regression and GLM. Comment on the most important aspects of this fitting process.

Solution: a) Two solutions, using Python or R

Python: Assuming you have your data in *crab_data_features* and targets in *crab_data_targets*

```
clf = linear_model.LinearRegression()
clf.fit (crab_data_features, crab_data_targets)
# The Residual sum of squares
print("Residual sum of squares: %.6f" % np.sum((clf.predict(crab_data_features) - crab_data_targets) ** 2))
# The mean square error
print("The mean square error %.6f" % np.mean((clf.predict(crab_data_features) - crab_data_targets) ** 2))
```

R: Assuming you have the complete data (features and labels) in *crab_data*

```
lm.fit = lm(satellites~.,data = crab_data)
sm = summary(lm.fit)
sum(residuals(lm.fit)^2)
mean(sm$residuals^2)
```

In both cases you get:

residual sum of squares	mean square error
1451.904	8.392507

b) First, let us verify the correctness of the weight update rule for a loglinear link function with Poisson distribution:

Suppose that $p(y|\mathbf{x}) \sim$Poisson$\left(\lambda = e^{\mathbf{w}^T\mathbf{x}}\right)$. Then:

$$p(y|\mathbf{x}) \quad = \quad \frac{\lambda^y e^{-\lambda}}{y!} = \frac{\left(e^{\mathbf{w}^T\mathbf{x}}\right)^y e^{e^{-\mathbf{w}^T\mathbf{x}}}}{y!} = \frac{e^{y\mathbf{w}^T\mathbf{x}} e^{e^{-\mathbf{w}^T\mathbf{x}}}}{y!} \qquad \text{for } y \in \mathbb{N}$$

As usual, assuming that observations $\mathbf{x}$ are independent, we can form the likelihood:

$$l(\mathbf{w}) = \prod_{i=1}^{n} p(y_i|\mathbf{x_i}) = \prod_{i=1}^{n} \frac{e^{y_i \mathbf{w}^T\mathbf{x}_i} e^{e^{-\mathbf{w}^T\mathbf{x}_i}}}{y_i!}$$

Clearly, it is easier to work with the log-likelihood:

$$\begin{aligned} ll(\mathbf{w}) &= \log\left(\prod_{i=1}^{n} \frac{e^{y_i \mathbf{w}^T \mathbf{x}_i} e^{e^{-\mathbf{w}^T \mathbf{x}_i}}}{y_i!}\right) \\ &= \sum_{i=1}^{n} \log\left(\frac{e^{y_i \mathbf{w}^T \mathbf{x}_i} e^{e^{-\mathbf{w}^T \mathbf{x}_i}}}{y_i!}\right) \\ &= \sum_{i=1}^{n} \log\left(e^{y_i \mathbf{w}^T \mathbf{x}_i}\right) + \log\left(e^{e^{-\mathbf{w}^T \mathbf{x}_i}}\right) - \log(y_i!) \\ &= \sum_{i=1}^{n} y_i \mathbf{w}^T \mathbf{x}_i - e^{\mathbf{w}^T \mathbf{x}_i} - \log(y_i!) \end{aligned}$$

Optimize:

$$\begin{aligned} \frac{\partial ll(\mathbf{w})}{\partial w_j} &= \frac{\partial}{\partial w_j}\left[\sum_{i=1}^{n} y_i \mathbf{w}^T \mathbf{x}_i - e^{\mathbf{w}^T \mathbf{x}_i} - \log(y_i!)\right] \\ &= \sum_{i=1}^{n} y_i \frac{\partial}{\partial w_j}\left[\mathbf{w}^T \mathbf{x}_i\right] - \frac{\partial}{\partial w_j}\left[e^{\mathbf{w}^T \mathbf{x}_i}\right] - \frac{\partial}{\partial w_j}\left[\log(y_i!)\right] \\ &= \sum_{i=1}^{n} y_i x_{ij} - x_{ij} e^{\mathbf{w}^T \mathbf{x}_i} \\ &= \sum_{i=1}^{n} x_{ij}\left(y_i - e^{\mathbf{w}^T \mathbf{x}_i}\right) \end{aligned}$$

Since $\mathbf{w}$ depends on $\mathbf{x}_i$, we cannot solve $\nabla(ll) = 0$ this analytically. Hence, we use Newton-Raphson:

$$\begin{aligned} \frac{\partial ll^2(\mathbf{w})}{\partial w_j \partial w_k} &= \frac{\partial}{\partial w_k}\left[\sum_{i=1}^{n} x_{ij}\left(y_i - e^{\mathbf{w}^T \mathbf{x}_i}\right)\right] \\ &= \sum_{i=1}^{n} x_{ij} \frac{\partial}{\partial w_k}\left[\left(y_i - e^{\mathbf{w}^T \mathbf{x}_i}\right)\right] \\ &= \sum_{i=1}^{n} -x_{ij} x_{ik} e^{\mathbf{w}^T \mathbf{x}_i} \end{aligned}$$

This verifies the correctness of the weight update rule for a loglinear link function with Poisson distribution. Using the notation of the notes, the method can be written as:

$$\boxed{\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \left(\mathbf{X}^T \mathbf{C}^{(t)} \mathbf{X}\right)^{-1} \mathbf{X}^T (\mathbf{y} - \mathbf{c}^{(t)})}$$

where we can pick the initial weights $\mathbf{w}^{(0)}$ either randomly or to be all zeros.

R has a convenient implementation of this generalized linear model. The following code implements the method, assuming you have the data on a variable called *crab_data*:

```
model=glm(satellites~.,family=poisson(link=log),data = crab_data)
sm = summary(model)
sum(residuals(model)^2)
mean(model$residuals^2)
```

Where the parameter (*satellites* $\sim$.)of the function *glm* indicates that we want to predict the "satellites" row given all other features. The following table summarizes the results we want:

residual sum of squares	mean square error
551.1314	1.290214

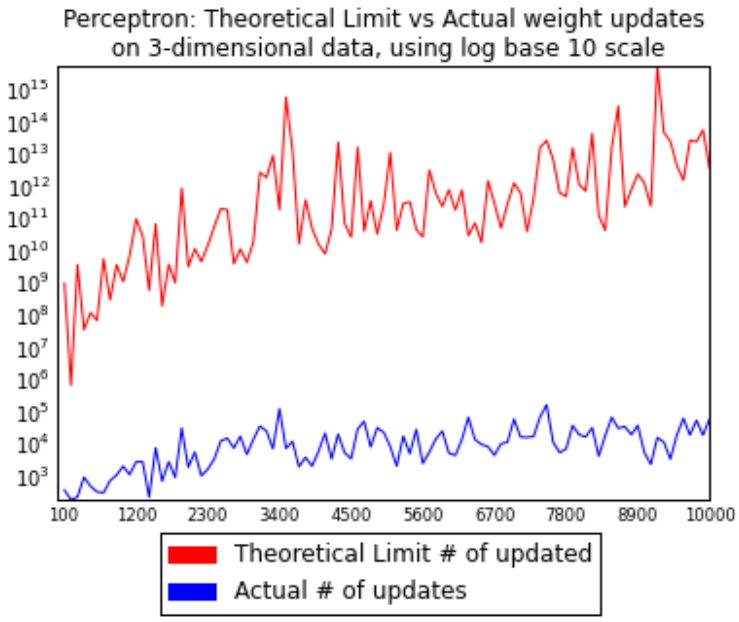
Note that the Generalized Linear model with a loglinear link function and Poisson distribution fits the crab data much better that an Ordinary Linear Model. This is evidence in support of the claim that, because the target variable (number of male satellites) is a non-negative integer, it may be beneficial to model it by a Poisson distribution, instead of Gaussian that is a base distribution in OLS linear regression. The following table compares the fit for both GLS and OLS:

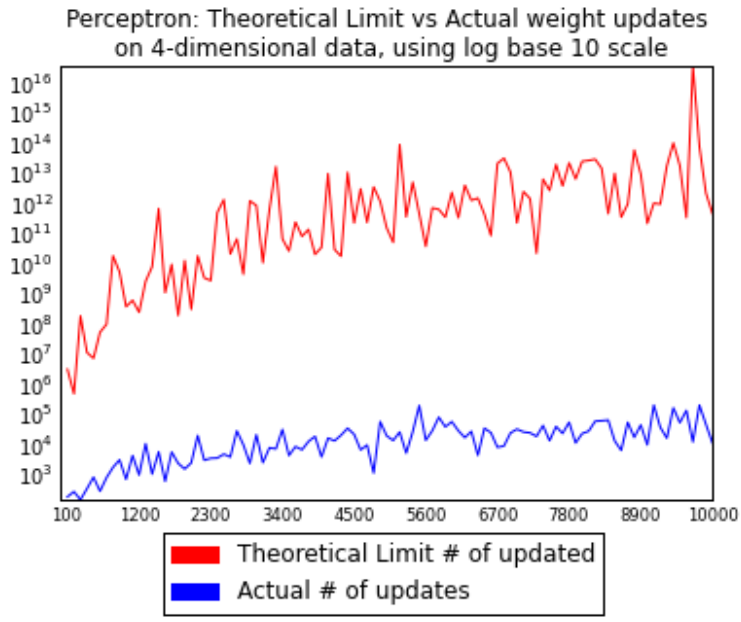
	OLS	GLS
residual sum of squares	1451.904	551.1314
mean square error	8.392507	1.290214

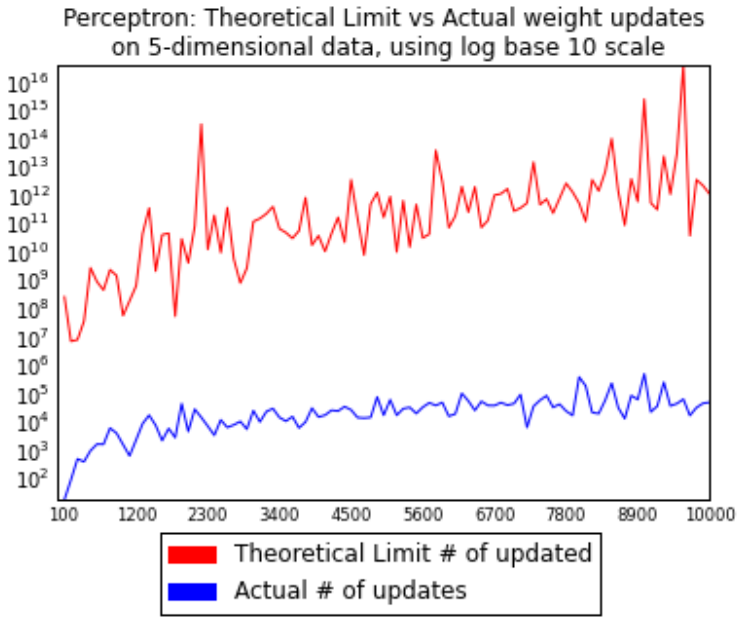
Appendix

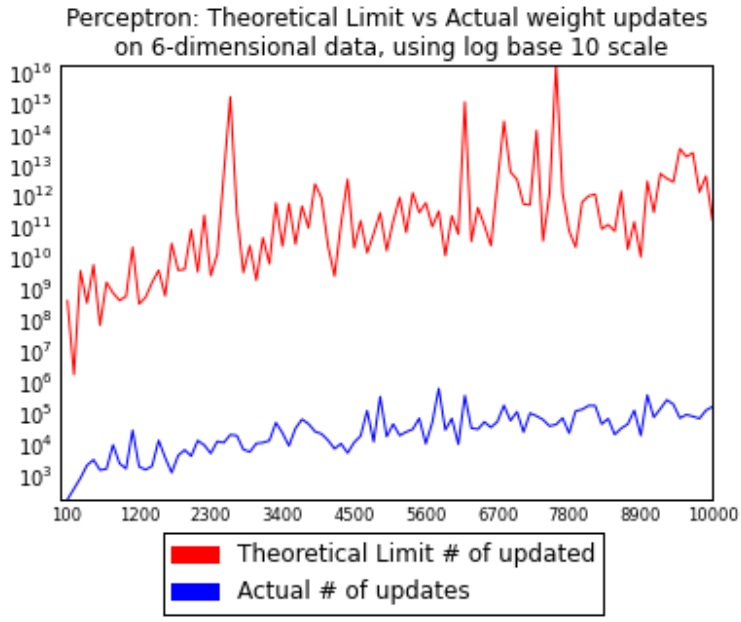
Appendix to Problem 4, part a). The following charts correspond to the results of problem 4, part a). Here you will find 8 plots, for dimension $d = 3, 4, \ldots, 10$, comparing the theoretical limit on the weights updates on a Perceptron against the actual number of updates. Points on the red line correspond to theoretical limit for different data sizes as labeled on the x-axis and points on the blue line correspond to actual weight updates. The idea is that for a given data set we compute the theoretical limit and the actual number of updates obtaining two points, one red and one blue. Points are connected with a line for visual aid. Points on the plot correspond to results on different data sizes from $N = 100, 200, \ldots, 10,000$.

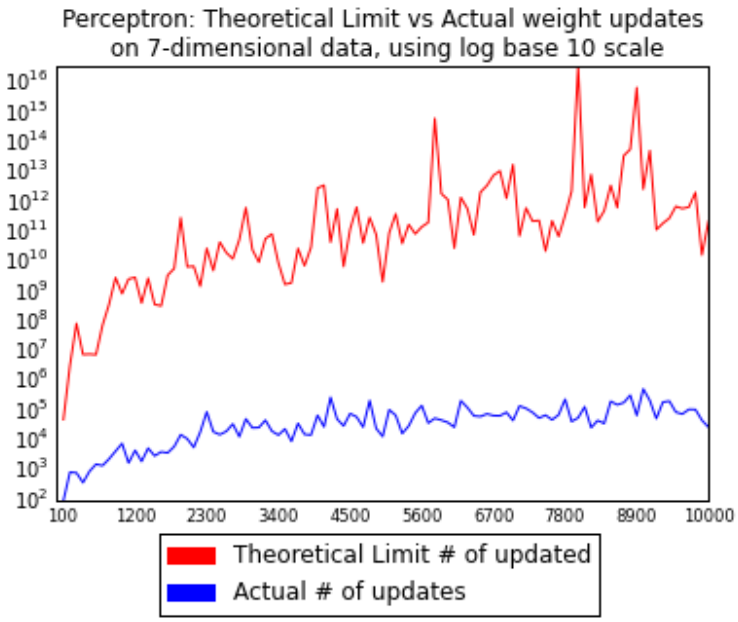
Note that the data was produced to be linearly separable from a $d-$dimensional Guassian distribution. The graphs provide strong evidence that the theoretical limit is a very loose bound on the actual number of updates. In general the algorithm requires considerable fewer weight updates to converge.

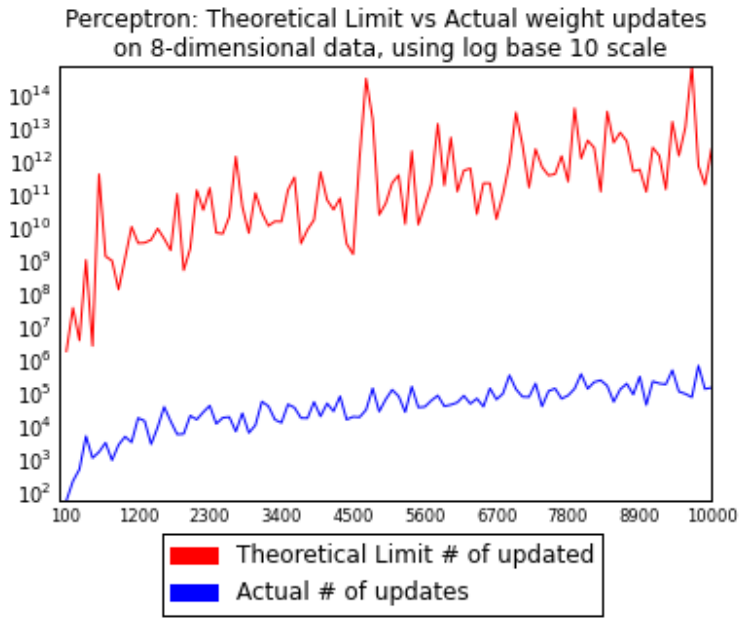


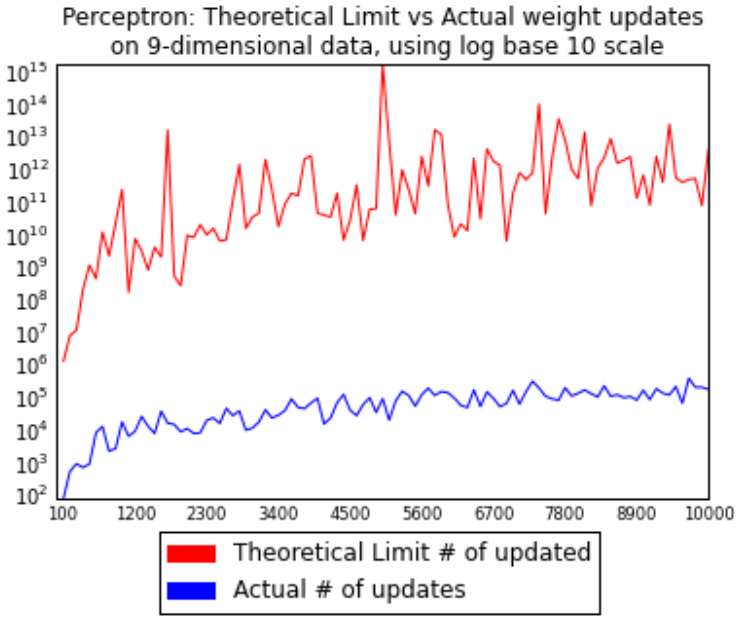


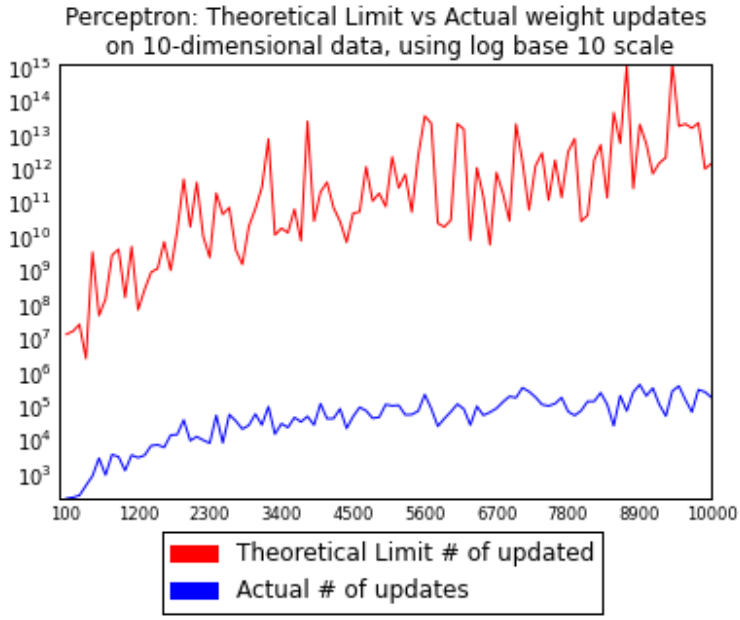












Note that all of these graphs are fairly similar suggesting that the way data was generated does not influence the number of theoretical and actual weight updates for Perceptron. This is of course a limited test on this topic with a particular way of generating data. Other data generation methods might show a more significant different among data of different dimensions. However, the main result, i.e., that the theoretical limit is a loose bound remains valid throughout all of these data.

References

[1.] Instructor's note.

[2.] https://onlinecourses.science.psu.edu/stat504/node/169

[3.] http://www.cs.rit.edu/ rlaz/PatternRecognition/slides/Bayesian.pdf

[4.] http://www.cse.buffalo.edu/ jcorso/t/2011S_555/files/homework1-solutions.pdf

[5.] http://www.randalolson.com/2014/06/28/how-to-make-beautiful-data-visualizations-in-python-with-matplotlib/