ECE 595: Homework 4

Yi Qiao, Class ID 187

(Spring 2019)

Exercise 1: Theory

(a) Logistic regression

(i)

$$\underset{\boldsymbol{\theta}}{argmin} \sum_{j=1}^{N} -\{y_j \log h_{\boldsymbol{\theta}}(\boldsymbol{x}_j) + (1-y_j)\log(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}_j))\}$$

$$= \underset{\boldsymbol{\theta}}{argmin} \sum_{j=1}^{N} -\left\{\log(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}_j)) + y_j \log \frac{h_{\boldsymbol{\theta}}(\boldsymbol{x}_j)}{1-h_{\boldsymbol{\theta}}(\boldsymbol{x}_j)}\right\}$$

$$= \underset{\boldsymbol{\theta}}{argmin} \sum_{j=1}^{N} -\left\{\log(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}_j)) + y_j \log \frac{\frac{1}{1+e^{-\boldsymbol{\theta}^T\boldsymbol{x}_j}}}{\frac{e^{-\boldsymbol{\theta}^T\boldsymbol{x}_j}}{1+e^{-\boldsymbol{\theta}^T\boldsymbol{x}_j}}}\right\} \tag{1}$$

$$= \underset{\boldsymbol{\theta}}{argmin} \sum_{j=1}^{N} -\left\{\log(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}_j)) + y_j\boldsymbol{\theta}^T\boldsymbol{x}_j\right\}$$

Take the derivative..

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \sum_{j=1}^{N} -\left\{\log(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}_j)) + y_j\boldsymbol{\theta}^T\boldsymbol{x}_j\right\} &= 0 \\
\sum_{j=1}^{N} -\left\{\nabla_{\boldsymbol{\theta}}\log(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}_j)) + \nabla_{\boldsymbol{\theta}} y_j\boldsymbol{\theta}^T\boldsymbol{x}_j\right\} &= 0 \\
\sum_{j=1}^{N} \{\boldsymbol{x}_j h_{\boldsymbol{\theta}}(\boldsymbol{x}_j) - y_j\boldsymbol{x}_j\} &= 0 \\
\sum_{j=1}^{N} \{(h_{\boldsymbol{\theta}}(\boldsymbol{x}_j) - y_j)\,\boldsymbol{x}_j\} &= 0 \\
h_{\boldsymbol{\theta}}(\boldsymbol{x}_j) &= y_j
\end{aligned} \tag{2}$$

Which implies, when linearly separable...
When $y_j = 1$,

$$\boldsymbol{\theta}^T\boldsymbol{x}_j = \infty$$

When $y_j = 0$,

$$\boldsymbol{\theta}^T\boldsymbol{x}_j = -\infty$$

Since x_j is always bounded, obviously, $\boldsymbol{\theta}$ tends to ∞.

Since $\boldsymbol{\theta}$ goes to ∞ for $\nabla_{\boldsymbol{\theta}}$ goes to 0, for any finite $\boldsymbol{\theta}$, the derivative

$$\sum_{j=1}^{N} \{(h_{\boldsymbol{\theta}}(\boldsymbol{x}_j) - y_j)\, \boldsymbol{x}_j\}$$

will never goes to 0. Thus the gradient decent,

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \alpha_k \left(\sum_{j=1}^{N} (h_{\boldsymbol{\theta}^{(k)}}(\boldsymbol{x}_j) - y_j)\, \boldsymbol{x}_j \right)$$

will never stop for any finite k.

(ii)

If linearly separable, the discriminant function will go as steep as possible. Thus, $\boldsymbol{w},\ w_0$ will go as far as possible.
We can also add some regularization to the original optimization problem, like

$$\underset{\boldsymbol{\theta}}{argmin} \sum_{j=1}^{N} -\{y_j \log h_{\boldsymbol{\theta}}(\boldsymbol{x}_j) + (1 - y_j) \log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}_j))\} + \lambda ||\boldsymbol{\theta}||^2$$

(iii)

No. Since no one else use log in the loss function.

(b) Perceptron

(i)

$$\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} + y_j \boldsymbol{x}_j,\ \textit{some } j \in \mathcal{M}_k$$

$$\begin{aligned} y_j(\boldsymbol{w}^{(k+1)})^T \boldsymbol{x}_j &= y_j(\boldsymbol{w}^{(k)} + y_j \boldsymbol{x}_j)^T \boldsymbol{x}_j \\ &= y_j(\boldsymbol{w}^{(k)})^T \boldsymbol{x}_j + y_j^2 \boldsymbol{x}_j^T \boldsymbol{x}_j \\ &= y_j(\boldsymbol{w}^{(k)})^T \boldsymbol{x}_j + y_j^2 ||\boldsymbol{x}_j||_2^2 \end{aligned} \tag{3}$$

The second term $y_j^2 ||\boldsymbol{x}_j||_2^2 \geq 0$, thus,

$$y_j(\boldsymbol{w}^{(k+1)})^T \boldsymbol{x}_j > y_j(\boldsymbol{w}^{(k)})^T \boldsymbol{x}_j$$

Thus making the loss function smaller.

(ii)

Step 1. Show that $(\boldsymbol{w}^{(k)})^T\boldsymbol{w}^* \geq k\rho$
Assuming $\boldsymbol{w}^{(0)} = 0$, obviously, $(\boldsymbol{w}^{(0)})^T\boldsymbol{w}^* = 0 \geq 0\rho$
Assuming $(\boldsymbol{w}^{(k-1)})^T\boldsymbol{w}^* \geq (k-1)\rho$,

$$\begin{aligned}(\boldsymbol{w}^{(k)})^T\boldsymbol{w}^* &= (\boldsymbol{w}^{(k-1)} + y_j\boldsymbol{x}_j)^T\boldsymbol{w}^*,\ \textit{some } j \in \mathcal{M}_k \\ &= (\boldsymbol{w}^{(k-1)})^T\boldsymbol{w}^* + y_j\boldsymbol{x}_j^T\boldsymbol{w}^* \\ &\geq (\boldsymbol{w}^{(k-1)})^T\boldsymbol{w}^* + \min_j y_j\boldsymbol{x}_j^T\boldsymbol{w}^* \\ &\geq (\boldsymbol{w}^{(k-1)})^T\boldsymbol{w}^* + \rho \\ &\geq k\rho\end{aligned} \tag{4}$$

By induction, $(\boldsymbol{w}^{(k)})^T\boldsymbol{w}^* \geq k\rho$ is proved.
Step 2. Show that $||\boldsymbol{w}^{(k)}||_2^2 \leq ||\boldsymbol{w}^{(k-1)}||_2^2 + ||\boldsymbol{x}_j||_2^2$

$$\begin{aligned}||\boldsymbol{w}^{(k)}||_2^2 &= ||\boldsymbol{w}^{(k-1)} + y_j\boldsymbol{x}_j||_2^2,\ \textit{some } j \in \mathcal{M}_{k-1} \\ &= ||\boldsymbol{w}^{(k-1)}||_2^2 + y_j^2||\boldsymbol{x}_j||_2^2 + 2y_j(\boldsymbol{w}^{(k-1)})^T x_j 1,\ \textit{some } j \in \mathcal{M}_{k-1}\end{aligned} \tag{5}$$

Since $j \in \mathcal{M}_{k-1}$, $y_j(\boldsymbol{w}^{(k-1)})^T\boldsymbol{x}_j < 0$, thus,

$$||\boldsymbol{w}^{(k)}||_2^2 \leq ||\boldsymbol{w}^{(k-1)}||_2^2 + ||\boldsymbol{x}_j||_2^2$$

is proved.
Also,

$$\begin{aligned}||\boldsymbol{w}^{(k)}||_2^2 &\leq ||\boldsymbol{w}^{(k-1)}||_2^2 + \max_j ||\boldsymbol{x}_j||_2^2 \\ &= ||\boldsymbol{w}^{(k-1)}||_2^2 + R^2\end{aligned} \tag{6}$$

By induction, we get,

$$||\boldsymbol{w}^{(k)}||_2^2 \leq kR^2$$

Step 3.
Using 1. and 2., by inspection,

$$\begin{aligned}\frac{(\boldsymbol{w}^{(k)})^T\boldsymbol{w}^*}{||\boldsymbol{w}^{(k)}||_2} &\geq \frac{k\rho}{\sqrt{kR^2}} \\ &= \sqrt{k}\frac{\rho}{R}\end{aligned} \tag{7}$$

Which implies,

$$\begin{aligned}\sqrt{k} &\leq \frac{R(\boldsymbol{w}^{(k)})^T\boldsymbol{w}^*}{\rho||\boldsymbol{w}^{(k)}||_2} \\ k &\leq \frac{R^2(\boldsymbol{w}^{(k)})^T(\boldsymbol{w}^{(k)})\boldsymbol{w}^{*T}\boldsymbol{w}^*}{\rho^2||\boldsymbol{w}^{(k)}||_2^2} \\ &\leq \frac{R^2||\boldsymbol{w}^*||_2^2}{\rho^2}\end{aligned} \tag{8}$$

(c) SVM

(i)

$$\begin{aligned} &\underset{\boldsymbol{\theta},\boldsymbol{\xi}}{argmin}\ \frac{1}{2}||\boldsymbol{w}||_2^2 + C||\boldsymbol{\xi}||_2^2 \\ &subject\ to\ y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j) \geq 1 - \xi_j,\ \xi_j \geq 0\ \forall j \end{aligned}$$

By taking the derivative, we get,

$$\begin{aligned} \nabla_{\boldsymbol{\xi}} \frac{1}{2}||\boldsymbol{w}||_2^2 + C||\boldsymbol{\xi}||_2^2 &= 2C\boldsymbol{\xi} \\ \nabla_{\boldsymbol{\theta}} \frac{1}{2}||\boldsymbol{w}||_2^2 + C||\boldsymbol{\xi}||_2^2 &= \boldsymbol{w} \end{aligned} \tag{9}$$

When $2C\boldsymbol{\xi} = 0$, obviously $\xi_j \geq 0$ is satisfied.
When $\boldsymbol{w} = 0$, $0 \geq 1 - \xi_j$, which implies $\xi_j \geq 1$.
No matter bounded by which one, $\xi_j \geq 0$ is satisfied, thus such condition has no effect on the solution.

(ii)

$$\begin{aligned} &\underset{\boldsymbol{\theta},\boldsymbol{\xi}}{argmin}\ \frac{1}{2}||\boldsymbol{w}||_2^2 + \frac{1}{2}C||\boldsymbol{\xi}||_2^2 \\ &subject\ to\ y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j) \geq 1 - \xi_j, j = 1, .., N \end{aligned} \tag{10}$$

Writing out the Lagrangian,

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\xi}, \boldsymbol{\mu}) &= \frac{1}{2}||\boldsymbol{w}||_2^2 + \frac{1}{2}C||\boldsymbol{\xi}||_2^2 - \sum_{j=1}^{N} \mu_j(\xi_j - 1 + y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j)) \\ &= \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{1}{2}C\boldsymbol{\xi}^T\boldsymbol{\xi} + \sum_{j=1}^{N} \mu_j(1 - y_j w_0) + \sum_{j=1}^{N} -\mu_j\xi_j + \sum_{j=1}^{N} -\mu_j y_j \boldsymbol{w}^T\boldsymbol{x}_j \\ &= (\frac{1}{2}\boldsymbol{w} - \sum_{j=1}^{N} \mu_j y_j \boldsymbol{x}_j)^T\boldsymbol{w} + (\frac{1}{2}C\boldsymbol{\xi} - \boldsymbol{\mu})^T\boldsymbol{\xi} + \sum_{j=1}^{N} \mu_j(1 - y_j w_0) \\ \nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{\mu}) &= \boldsymbol{w} - \sum_{j=1}^{N} \mu_j y_j \boldsymbol{x}_j = 0 \\ \nabla_{w_0}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\xi}, \boldsymbol{\mu}) &= \sum_{j=1}^{N} -\mu_j y_j = 0 \\ \nabla_{\boldsymbol{\xi}}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\xi}, \boldsymbol{\mu}) &= C\boldsymbol{\xi} - \boldsymbol{\mu} = 0 \end{aligned} \tag{11}$$

Thus,

$$\begin{aligned} \boldsymbol{w}^* &= \sum_{j=1}^{N} \mu_j y_j \boldsymbol{x}_j \\ \sum_{j=1}^{N} \mu_j y_j &= 0 \\ C\xi_j^* &= \mu_j\ \forall j \end{aligned} \tag{12}$$

(iii)

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\xi}, \boldsymbol{\mu}) = \frac{1}{2}||\boldsymbol{w}||_2^2 + \frac{1}{2}C||\boldsymbol{\xi}||_2^2 - \sum_{j=1}^{N} \mu_j(\xi_j - 1 + y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j)) \tag{13}$$

Complementarity Conditions says,

$$\begin{aligned} \mu_j > 0 \; and \; \xi_j - 1 + y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j) = 0 \\ \mu_j = 0 \; and \; \xi_j - 1 + y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j) > 0 \end{aligned} \tag{14}$$

Which means,

$$\nabla_{\boldsymbol{\mu}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\xi}, \boldsymbol{\mu}) = \mathbf{0} \Rightarrow \underset{\boldsymbol{\mu} \geq \mathbf{0}}{argmax} \left\{ -\sum_{j=1}^{N} \mu_j(\xi_j - 1 + y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j)) \right\} \tag{15}$$

Then, the primal problem becomes,

$$\begin{aligned} \underset{\boldsymbol{\theta}, \boldsymbol{\xi}}{argmin} \; \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\xi}, \boldsymbol{\mu}^*) &= \underset{\boldsymbol{\theta}, \boldsymbol{\xi}}{argmin} \left\{ \frac{1}{2}||\boldsymbol{w}||_2^2 + \frac{1}{2}C||\boldsymbol{\xi}||_2^2 + \underset{\boldsymbol{\mu} \geq \mathbf{0}}{argmax} \left\{ -\sum_{j=1}^{N} \mu_j(\xi_j - 1 + y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j)) \right\} \right\} \\ &= \min_{\boldsymbol{\theta}, \boldsymbol{\xi}} \max_{\boldsymbol{\mu} \geq \mathbf{0}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\xi}, \boldsymbol{\mu}) \end{aligned} \tag{16}$$

The dual problem is,

$$\max_{\boldsymbol{\mu} \geq \mathbf{0}} \min_{\boldsymbol{\theta}, \boldsymbol{\xi}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\xi}, \boldsymbol{\mu}) = \max_{\boldsymbol{\mu} \geq \mathbf{0}} \left\{ \min_{\boldsymbol{\theta}, \boldsymbol{\xi}} \frac{1}{2}||\boldsymbol{w}||_2^2 + \frac{1}{2}C||\boldsymbol{\xi}||_2^2 - \sum_{j=1}^{N} \mu_j(\xi_j - 1 + y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j)) \right\} \tag{17}$$

Plug in solution we got in (ii),

$$\begin{aligned} &= \max_{\boldsymbol{\mu} \geq \mathbf{0}} \left\{ \frac{1}{2}||\sum_{j=1}^{N} \mu_j y_j \boldsymbol{x}_j||_2^2 + \frac{1}{2}C||\frac{\boldsymbol{\mu}}{C}||_2^2 - \sum_{j=1}^{N} \mu_j(\frac{\mu_j}{C} - 1 + y_j(\sum_{j=1}^{N} \mu_j y_j \boldsymbol{x}_j^T \boldsymbol{x}_j + w_0)) \right\} \\ &= \max_{\boldsymbol{\mu} \geq \mathbf{0}} \left\{ \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \mu_i \mu_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j + \frac{1}{2C} \boldsymbol{\mu}^T \boldsymbol{\mu} - \sum_{j=1}^{N} \frac{\mu_j^2}{C} + \sum_{j=1}^{N} \mu_j - \sum_{j=1}^{N} \mu_j y_j (\sum_{j=1}^{N} \mu_j y_j \boldsymbol{x}_j^T \boldsymbol{x}_j + w_0)) \right\} \\ &= \max_{\boldsymbol{\mu} \geq \mathbf{0}} \left\{ \sum_{j=1}^{N} \mu_j + \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \mu_i \mu_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j - \frac{1}{2} \sum_{j=1}^{N} \frac{\mu_j^2}{C} \right\} \; subject \; to \; \sum_{j=1}^{N} \mu_j y_j = 0 \end{aligned} \tag{18}$$

(d) SVM

(i)

$$\begin{aligned} &\underset{\boldsymbol{\theta}}{argmin}\ \frac{1}{2}||\boldsymbol{w}||_2^2 \\ &subject\ to\ y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j) \geq \gamma, j = 1, ..., N \end{aligned} \tag{19}$$

Scaling the subjective function,

$$y_j(\frac{\boldsymbol{w}^T}{\gamma}\boldsymbol{x}_j + \frac{w_0}{\gamma}) \geq 1$$

Substituting $\frac{\boldsymbol{w}^T}{\gamma} \rightarrow \boldsymbol{w}^T$, $\frac{w_0}{\gamma} \rightarrow w_0$, The problem becomes

$$\begin{aligned} &\underset{\gamma\boldsymbol{\theta}}{argmin}\ \frac{\gamma^2}{2}||\boldsymbol{w}||_2^2 \\ &subject\ to\ y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j) \geq 1, j = 1, ..., N \end{aligned} \tag{20}$$

which is the same as,

$$\begin{aligned} &\underset{\gamma\boldsymbol{\theta}}{argmin}\ \frac{1}{2}||\boldsymbol{w}||_2^2 \\ &subject\ to\ y_j g_{\boldsymbol{\theta}}(\boldsymbol{x}_j) \geq 1, j = 1, ..., N \end{aligned} \tag{21}$$

By inspection, the solution will be

$$\boldsymbol{\theta}^* = \frac{1}{\gamma}\boldsymbol{\theta}^*_{original}$$

Since the decision boundary is,

$$\boldsymbol{w}^T\boldsymbol{x} + w_0 = 0$$

multiply $\boldsymbol{\theta}$ by some scalar $\frac{1}{\gamma}$, the new decision boundary becomes,

$$\frac{1}{\gamma}\boldsymbol{w}^T\boldsymbol{x} + \frac{1}{\gamma}w_0 = 0$$

which is the same as the original decision boundary.

(ii)

Suppose we have two data points, $\{\boldsymbol{x}_0, -1\}$ and $\{\boldsymbol{x}_1, 1\}$, the Hard-Margin SVM problem is:

$$\begin{aligned} &\underset{\boldsymbol{\theta}}{argmin}\ \frac{1}{2}||\boldsymbol{w}||_2^2 \\ &subject\ to\ -(\boldsymbol{w}^T\boldsymbol{x}_0 + w_0) \geq 1,\ \boldsymbol{w}^T\boldsymbol{x}_1 + w_0 \geq 1 \end{aligned} \tag{22}$$

Writing out the Lagrangian,

$$\begin{aligned} \mathcal{L}(\boldsymbol{w}, w_0, \boldsymbol{\mu}) &= \frac{1}{2}||\boldsymbol{w}||_2^2 - \mu_0(-(\boldsymbol{w}^T\boldsymbol{x}_0 + w_0) - 1) - \mu_1(\boldsymbol{w}^T\boldsymbol{x}_1 + w_0 - 1) \\ \nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w}, w_0, \boldsymbol{\mu}) &= \boldsymbol{w} + \mu_0\boldsymbol{x}_0 - \mu_1\boldsymbol{x}_1 = 0 \\ \nabla_{w_0}\mathcal{L}(\boldsymbol{w}, w_0, \boldsymbol{\mu}) &= \mu_0 - \mu_1 = 0 \end{aligned} \tag{23}$$

The primal problem is,

$$\underset{\boldsymbol{w}, w_0}{argmin}\ \mathcal{L}(\boldsymbol{w}, w_0, \boldsymbol{\mu}^*) = \min_{\boldsymbol{w}, w_0}\left\{\frac{1}{2}||\boldsymbol{w}||_2^2 + \max_{\boldsymbol{\mu}\geq\boldsymbol{0}}\left\{-\mu_0(-(\boldsymbol{w}^T\boldsymbol{x}_0 + w_0) - 1) - \mu_1(\boldsymbol{w}^T\boldsymbol{x}_1 + w_0 - 1)\right\}\right\} \tag{24}$$

The dual problem is,

$$\begin{aligned} &\max_{\boldsymbol{\mu}\geq\boldsymbol{0}}\min_{\boldsymbol{w}, w_0}\left\{\frac{1}{2}||\boldsymbol{w}||_2^2 + \left\{-\mu_0(-(\boldsymbol{w}^T\boldsymbol{x}_0 + w_0) - 1) - \mu_1(\boldsymbol{w}^T\boldsymbol{x}_1 + w_0 - 1)\right\}\right\} \\ &= \max_{\boldsymbol{\mu}\geq\boldsymbol{0}}\min_{\boldsymbol{w}, w_0}\left\{\frac{1}{2}||\mu_1\boldsymbol{x}_1 - \mu_0\boldsymbol{x}_0||_2^2 + \mu_0(\boldsymbol{w}^T\boldsymbol{x}_0 + w_0) + \mu_0 - \mu_1(\boldsymbol{w}^T\boldsymbol{x}_1 + w_0) + \mu_1\right\} \\ &= \max_{\boldsymbol{\mu}\geq\boldsymbol{0}}\left\{\frac{1}{2}\mu_0^2(\boldsymbol{x}_1 - \boldsymbol{x}_0)^T(\boldsymbol{x}_1 - \boldsymbol{x}_0) - \mu_0^2(\boldsymbol{x}_1 - \boldsymbol{x}_0)^T(\boldsymbol{x}_1 - \boldsymbol{x}_0) + 2\mu_0\right\} \\ &= \max_{\boldsymbol{\mu}\geq\boldsymbol{0}}\left\{-\frac{1}{2}\mu_0^2(\boldsymbol{x}_1 - \boldsymbol{x}_0)^T(\boldsymbol{x}_1 - \boldsymbol{x}_0) + 2\mu_0\right\} \end{aligned} \tag{25}$$

Taking derivative,

$$\begin{aligned} &\frac{\partial}{\partial\mu_0}\left\{-\frac{1}{2}\mu_0^2(\boldsymbol{x}_1 - \boldsymbol{x}_0)^T(\boldsymbol{x}_1 - \boldsymbol{x}_0) + 2\mu_0\right\} \\ &= -\mu_0(\boldsymbol{x}_1 - \boldsymbol{x}_0)^T(\boldsymbol{x}_1 - \boldsymbol{x}_0) + 2 = 0 \end{aligned} \tag{26}$$

which implies,

$$\begin{aligned} \mu_0^* = \mu_1^* &= \frac{2}{(\boldsymbol{x}_1 - \boldsymbol{x}_0)^T(\boldsymbol{x}_1 - \boldsymbol{x}_0)} \\ \boldsymbol{w}^* = \mu_0(\boldsymbol{x}_1 - \boldsymbol{x}_0) &= \frac{2}{(\boldsymbol{x}_1 - \boldsymbol{x}_0)^T(\boldsymbol{x}_1 - \boldsymbol{x}_0)}(\boldsymbol{x}_1 - \boldsymbol{x}_0) \\ w_0^* = -\frac{(\boldsymbol{x}_0 + \boldsymbol{x}_1)^T\boldsymbol{w}^*}{2} &= -\frac{(\boldsymbol{x}_0 + \boldsymbol{x}_1)^T(\boldsymbol{x}_1 - \boldsymbol{x}_0)}{(\boldsymbol{x}_1 - \boldsymbol{x}_0)^T(\boldsymbol{x}_1 - \boldsymbol{x}_0)} \end{aligned} \tag{27}$$

Exercise 2: Implementations

Import data

refer to code.

(a) Logistic Regression

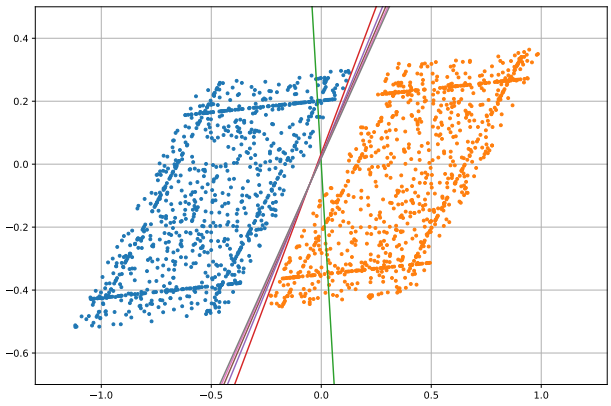
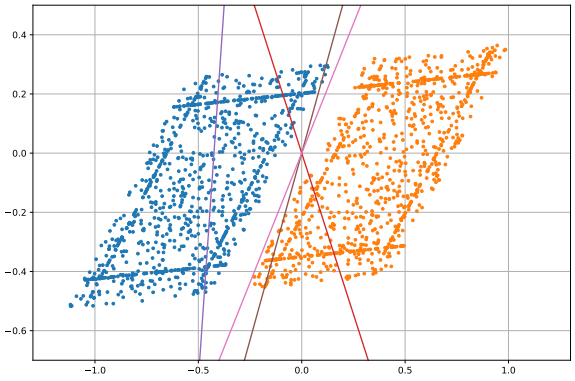


Figure 1: Gradient Decent of Logistic Regression

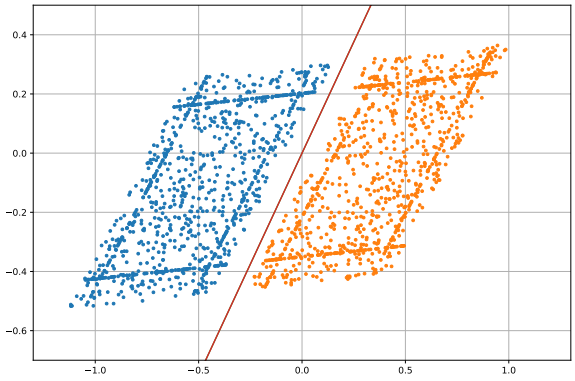
Increasing the maximum number of iterates will let the decision boundary converge to the desired boundary, after the loss is 0, the boundary will not change anymore.

Increasing the learning rate will change the step size, a learning rate that is too large may leads to oscillation and one that is too small may leads to slow convergence.

(b) Perceptron



(a) (i)Perceptron Online Mode

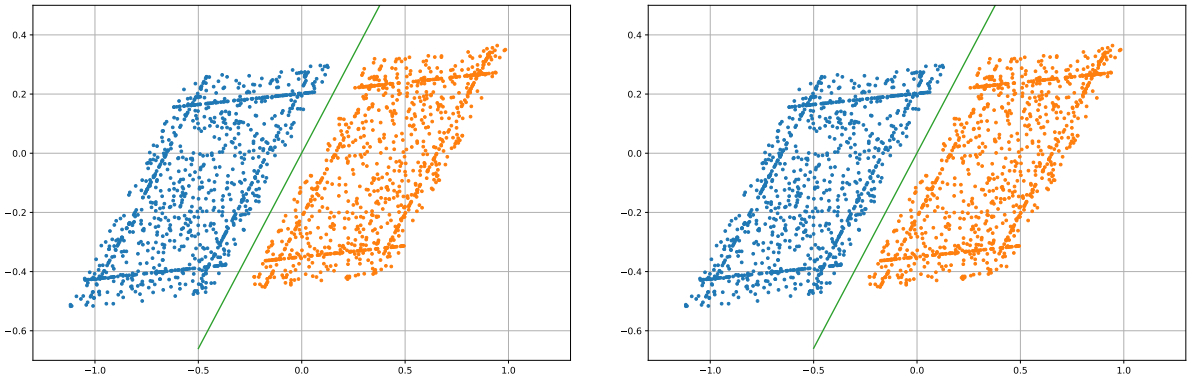


(b) (ii)Perceptron Batch Mode

Figure 2: Decision of Perceptron algorithm

Batch mode converge faster in term of rounds. However, the cost each round is a lot higher than the online mode.

(c) SVM



(a) (i)Hard SVM

(b) (ii)Soft SVM

Figure 3: Decision boundary of SVM

In this case, the final decision boundary for the soft case is no better than the hard case, since the data is linearly separable.

C is the strength of regularization. Since there is no data point actually goes inside the margin, C is useless in this case.

(d) Comments

In this case, since the data are linearly separable, the performance of each classifier is pretty much the same.

In terms of margin, perceptron is not as robust as the other two, in terms of providing a "good" decision boundary and from the graph we can see it has the smallest margin, also it is not a deterministic approach. The other both performs well, while SVM will obviously have larger margin.

However, larger margin does not necessarily means better performance, since over-fitting might be a problem here.

In all those classifiers, Soft SVM takes the longest to run.

Code

```
#! /usr/bin/env python3

import numpy as np
import cvxpy as cp
import matplotlib.pyplot as plt


def newfig():
    fig = plt.figure(figsize=(9,6), dpi=300)
    ax = fig.add_subplot(111)
    ax.grid()
    return fig, ax


def final_adjust(fn):
    plt.tight_layout()
    plt.savefig(fn, bbox='tight')


def drawline(ax, theta):
    slope = -theta[0] / theta[1]
    intercept = -theta[2] / theta[1]

    x1 = [-0.5, 0.5]
    x2 = [-0.5 * slope+intercept, 0.5 * slope+intercept]

    ax.plot(x1, x2, '-')


def import_data():

    samples = np.array(np.loadtxt('../data/hw04_sample_vectors.csv',
    ↪  delimiter=','))
    labels = np.array(np.loadtxt('../data/hw04_labels.csv', delimiter=','))

    return samples, labels


def logistic(X, labels, learning_rate=0.1, max_num_iterations=200):

    def hypfunc(theta, X):
        return 1 / (1 + np.exp(-np.dot(theta, np.concatenate([X, [1]]).T)))


    def cost_function_derivative(X, labels, theta):
```

```
        return sum([(hypfunc(theta, X[i]) - labels[i])*np.concatenate([X[i],
        ↪  [1]]) for i in range(len(X))])


    theta = np.array([0,0,0], dtype=np.float64)

    fig, ax = newfig()
    lenl1 = int(sum(labels))
    lenl0 = len(labels) - lenl1

    ax.plot(X[:lenl0,0], X[:lenl0,1], '.')
    ax.plot(X[lenl1:,0], X[lenl1:,1], '.')
    ax.axis([-1.3,1.3,-0.7,0.5])
    for m in range(max_num_iterations):
        theta -= learning_rate*cost_function_derivative(X, labels, theta)
        if m % (max_num_iterations / 5) == 0:
            drawline(ax, theta)

    drawline(ax, theta)
    final_adjust('../pix/logistic.pdf')


def perceptron(X, labels, learning_rate=0.0001, max_num_iterations=25, batchMode
↪  = False):

    theta = np.array([0, 0.0001, 0], dtype=np.float64)

    fig, ax = newfig()
    lenl1 = int(sum(labels))
    lenl0 = len(labels) - lenl1

    ax.plot(X[:lenl0,0], X[:lenl0,1], '.')
    ax.plot(X[lenl1:,0], X[lenl1:,1], '.')
    ax.axis([-1.3,1.3,-0.7,0.5])

    for m in range(max_num_iterations):
        shuffled_index = np.random.permutation(labels.size)
        X = X[shuffled_index, :]
        labels = labels[shuffled_index]

        misclassified = False
        for i, label in enumerate(labels):
            if np.dot(theta, np.concatenate([X[i], [1]]).T) * (1.0 if label > 0
            ↪  else -1.0) < 0:
                theta += learning_rate * (1.0 if label>0 else -1.0) *
                ↪  np.concatenate([X[i], [1]])
                misclassified = True
                if not batchMode: break
```

```
            if m % (max_num_iterations / 5) == 0:
                drawline(ax, theta)

            if not misclassified:
                break

    drawline(ax, theta)
    fn = '../pix/perceptron_' + ('batch' if batchMode else 'online') + '.pdf'
    final_adjust(fn)


def hardsvm(X, labels):
    fig, ax = newfig()
    len1 = int(sum(labels))
    len0 = len(labels) - len1

    ax.plot(X[:len0,0], X[:len0,1], '.')
    ax.plot(X[len1:,0], X[len1:,1], '.')

    labels = labels.copy() * 2 - 1
    w = cp.Variable(2)
    w0 = cp.Variable(1)
    objective = cp.Minimize(cp.sum_squares(w))
    constraints = [labels[i] * (w * X[i].T + w0) >= 1 for i in range(len(X))]
    problem = cp.Problem(objective, constraints)
    problem.solve()

    print(w.value)
    print(w0.value)


    ax.axis([-1.3,1.3,-0.7,0.5])
    drawline(ax, np.concatenate([w.value, w0.value]))
    final_adjust('../pix/hardsvm.pdf')


def softsvm(X, labels, C=1):
    fig, ax = newfig()
    len1 = int(sum(labels))
    len0 = len(labels) - len1

    ax.plot(X[:len0,0], X[:len0,1], '.')
    ax.plot(X[len1:,0], X[len1:,1], '.')
    labels = labels.copy() * 2 - 1

    w = cp.Variable(2)
    w0 = cp.Variable(1)
```

```
    xi = cp.Variable(len(X))

    objective = cp.Minimize(cp.sum_squares(w)/2 + C*cp.norm(xi,1))
    constraints = [labels[i] * (w * X[i].T + w0) >= 1-xi[i] for i in
    ↪  range(len(X))]
    problem = cp.Problem(objective, constraints)
    problem.solve()


    ax.axis([-1.3,1.3,-0.7,0.5])
    drawline(ax, np.concatenate([w.value, w0.value]))
    final_adjust('../pix/softsvm.pdf')

if __name__ == "__main__":

    X, labels = import_data()
    #logistic(X, labels)
    #perceptron(X, labels)
    #perceptron(X, labels, batchMode=True)
    #hardsvm(X,labels)
    softsvm(X,labels)
```