Assignment 2

ECE656

Feb 27, 2018 Christopher Robbiano

Problem 1

Derive the learning rule for logistic regression with Gaussian PDF for $y = f(\mathbf{x}^T\mathbf{w}) + \varepsilon$, $\varepsilon \sim \mathcal{N}(0,1)$, $f(\mathbf{x}^T\mathbf{w}) = \frac{1}{1+\exp(-\mathbf{x}^T\mathbf{w})}$ and $p(y|\mathbf{x};\mathbf{w}) \sim \mathcal{N}\big(f(\mathbf{x}^T\mathbf{w}), 1\big)$

Solution.
The likelihood function is given by

$$p\big(y(k)|\mathbf{x}(k);\mathbf{w}(k)\big) = \frac{1}{\sqrt{2\pi}}\exp\left(\frac{1}{2}\big(d(k) - f(\mathbf{w}^T(k)\mathbf{x}(k)\big)^T\big(d(k) - f(\mathbf{w}^T(k)\mathbf{x}(k))\big)\right)$$

We stack all $y(k)$ into vector $\mathbf{y}$ and evaluate the log-likelihood function $\ln\big(p(\mathbf{y}|\mathbf{X};\mathbf{w}(k))\big) = \ln\Big(\prod_{k=0}^{N-1} p\big(y(k)|\mathbf{x}(k);\mathbf{w}(k)\big)\Big)$. The log-likelihood function is given by

$$\Lambda(k) = \ln(\frac{N}{\sqrt{2\pi}}) - \frac{1}{2}\sum_{i=0}^{N-1} y^2(i) - 2y(i)f\big(\mathbf{w}^T(k)\mathbf{x}(i)\big) + f^2\big(\mathbf{w}^T(k)\mathbf{x}(i)\big)$$

Taking the partial derivative of Λ with respect to $\mathbf{w}(k)$ evaluates to

$$\begin{aligned}\frac{\partial\Lambda(k)}{\partial\mathbf{w}(k)} &= -\frac{1}{2}\sum_{i=0}^{N-1} -2y(i)\mathbf{x}(i)f'\big(\mathbf{w}^T(k)\mathbf{x}(i)\big) + 2\mathbf{x}(i)f\big(\mathbf{w}^T(k)\mathbf{x}(i)\big)f'\big(\mathbf{w}^T(k)\mathbf{x}(i)\big)\\ &= -\frac{1}{2}\sum_{i=0}^{N-1}\big[-2y(i) + 2f\big(\mathbf{w}^T(k)\mathbf{x}(i)\big)\big]\mathbf{x}(i)f'\big(\mathbf{w}^T(k)\mathbf{x}(i)\big)\\ &= \sum_{i=0}^{N-1} e(i)\mathbf{x}(i)f'\big(\mathbf{w}^T(k)\mathbf{x}(i)\big)\end{aligned}$$

Plugging this result into the weight update for the logistic regression provides a results that is similar to the delta learning rule, except that this is a *batch learning* version, i.e.,

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mu\sum_{i=0}^{N-1} e(k)\mathbf{x}(i)f'\big(\mathbf{w}^T(k)\mathbf{x}(i)\big)$$

■

Problem 2

For the Hebbian learning rule with a linear neuron model, $o = \mathbf{w}^T\mathbf{x}$, show the following:

(a) The convergence property. Is this a stable learner? Why?

(b) That after one epoch, the accumulated weight adjustment is related to the sample correlation matrix of the data (ignoring high order terms in μ). This is why the learning rule is also called *Correlation Learning.*

(c) That you can arrive at this learning rule from the gradient descent algorithm and describe the associated cost function $J(\mathbf{w}(k))$.

Solution.

(a) Given the Hebbian learning rule using a linear neuron $\mathbf{w}(k+1) = \mathbf{w}(k) + \mu\mathbf{x}(k)o(k) = \mathbf{w}(k) + \mu\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{w}$, we can rewrite learning rule as

$$\mathbf{w}(k+1) = (\mathbf{I} + \mu\mathbf{x}(k)\mathbf{x}^T(k))\mathbf{w}(k) = \mathbf{A}\mathbf{w}(k)$$

Assuming that the Hebbian learning rule converges, we use Kushner's direct averaging method and letting $\mathbf{e}(k) = \mathbf{w}(k) - \mathbf{w}^*$ we can look at the mean difference between the kth weight update and the optimal weight $\mathbf{w}^*$

$$\begin{aligned} E[\mathbf{e}(k+1)] &= E[(\mathbf{I} + \mu\mathbf{x}(k)\mathbf{x}^T(k))\mathbf{e}(k)] \\ &= (\mathbf{I} + \mu\mathbf{R}_{xx})E[\mathbf{e}(k)] \end{aligned}$$

Expanding the relationship between $\mathbf{e}(k+1)$ and $\mathbf{e}(k)$ starting with $\mathbf{e}(0)$ gives the general form of $\mathbf{e}(n) = \mathbf{A}^n\mathbf{e}(0) \Rightarrow \mathbf{w}(n) - \mathbf{w}^* = \mathbf{A}^n(\mathbf{w}(0) - \mathbf{w}^*)$ and thus

$$\mathbf{w}(n) - \mathbf{w}^* = (\mathbf{I} + \mu\mathbf{R}_{xx})^n(\mathbf{w}(0) - \mathbf{w}^*)$$

Diagonalizing $\boldsymbol{R_{xx} = Q\Lambda Q^T}$ gives

$$\begin{aligned} \mathbf{w}(n) - \mathbf{w}^* &= (\mathbf{Q}\mathbf{Q}^T + \mu\mathbf{Q}\Lambda\mathbf{Q}^T)^n(\mathbf{w}(0) - \mathbf{w}^*) \\ &= \mathbf{Q}^n(\mathbf{I} + \mu\Lambda)^n\mathbf{Q}^{T^n}(\mathbf{w}(k) - \mathbf{w}^*) \\ \Rightarrow \mathbf{Q}^{T^n}(\mathbf{w}(n) - \mathbf{w}^*) &= (\mathbf{I} + \mu\Lambda)^n\mathbf{Q}^{T^n}(\mathbf{w}(0) - \mathbf{w}^*) \\ \Upsilon(n) &= (\mathbf{I} + \mu\Lambda)^n\Upsilon(0) \end{aligned}$$

where $\Upsilon(n) = \mathbf{Q}^{T^n}(\mathbf{w}(n) - \mathbf{w}^*)$. This is a state equation with no driving term, and $(\mathbf{I} + \mu\Lambda)$ must be nilpotent for stability (i.e. $||\Upsilon^n(n)|| \to 0$ as $n \to \infty$). Therefore, for $\mu > 0$ we get that $|1+\mu\lambda_i| < 1$ for all i. This is not possible since $\mathbf{R}_{xx}$ is a semi-definite matrix and thus the convergence property does not exist for Hebbian learning when using a linear neuron.

Examining the stability of a weight vector, we again assume that the Hebbian learning rule will converge to a final weight $\mathbf{w}^*$. This implies that $\Delta\mathbf{w} = \mathbf{0}$ and thus

$$\begin{aligned} \mathbf{w}^* &= \mathbf{w}^* + \Delta\mathbf{w} \\ &= \mathbf{w}^* + \mu\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{w}^* \\ \Rightarrow 0 &= \mu\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{w}^* \\ 0 &= E[\mu\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{w}^*] \\ 0 &= \mu\mathbf{R}_{xx}\mathbf{w}^* \\ \mu(\lambda\mathbf{w}^*) &= \mu(\mathbf{R}_{xx}\mathbf{w}^*) \\ \Rightarrow \lambda &= 0 \end{aligned}$$

i.e., the eigenvalue associated with $\mathbf{w}^*$ is 0. Since $\mathbf{R}_{xx} \geq 0$ there exists at least one eigenvalue that is greater than 0 and thus for some $\mathbf{w}^* + \boldsymbol{\varepsilon} \neq \mathbf{0}$ we get $\Delta\mathbf{w} > \mathbf{0}$. This shows that there is metastable convergence point at $\mathbf{w}^* = \mathbf{0}$ and any small perturbation to the $\mathbf{w}^*$ produces a change in the weights that will not converge back to $\mathbf{w}^* = \mathbf{0}$.

(b) Using the weight update equation $\mathbf{w}(k+1) = \mathbf{w}(k) + \mu\mathbf{x}(k)o(k)$, we can express the weight update equation after one iteration as follows

$$\begin{aligned}\mathbf{w}(1) &= \big(\mathbf{I} + \mu\mathbf{x}(0)\mathbf{x}^T(0)\big)\mathbf{w}(0)\\ \mathbf{w}(2) &= \big(\mathbf{I} + \mu\mathbf{x}(1)\mathbf{x}^T(1)\big)\mathbf{w}(1)\\ &= \mathbf{w}(0) + \mu\big[\sum_{k=0}^{1}\mathbf{x}(k)\mathbf{x}^T(k)\big] + \mu^2\big[\prod_{k=0}^{1}\mathbf{x}(k)\mathbf{x}^T(k)\big]\mathbf{w}(0)\end{aligned}$$

Ignoring higher order terms in μ we get that

$$\mathbf{w}(n) \approx \mathbf{w}(0) + \mu\hat{\mathbf{R}}_{xx}\mathbf{w}(0)$$

(c) Let the gradient descent weight update equation be represented as

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \frac{1}{2}\mu\nabla J(\mathbf{w}(k))$$

Comparing this to the weight update equation for the Hebbian learning we get the following relationship for $\nabla J(\mathbf{w}(k)$

$$\begin{aligned}\mu\mathbf{x}(k)o(k) &= -\frac{1}{2}\mu\nabla J(\mathbf{w}(k))\\ \Rightarrow \nabla J(\mathbf{w}(k)) &= -2\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{w}(k)\end{aligned}$$

Taking the anti-derivative of $\nabla J(\mathbf{w}(k))$ gives the following cost function for the gradient descent algorithm

$$\begin{aligned}J(\mathbf{w}(k)) &= -\mathbf{w}^T(k)\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{w}(k)\\ &= -||\mathbf{w}^T(k)\mathbf{x}(k)||^2\end{aligned}$$

■

Problem 3

Show that when using the Kohonen learning rule the weight vectors converge to the centroids of the clusters for arbitrary initial conditions. Comment on the convergence property of this learning. What are the factors that influence the convergence rate?

Solution. Using the metric $D(\mathbf{w}, \mathbf{x}) = ||\mathbf{x} - \mathbf{w}||^2$, the Kohonen update rule for the activated neuron is given by

$$\begin{aligned}
\mathbf{w}(k+1) &= \mathbf{w}(k) - \frac{\alpha}{2}\nabla D \\
&= \mathbf{w}(k) - \frac{\alpha}{2}\frac{\partial D}{\partial \mathbf{w}} \\
&= \mathbf{w}(k) - \frac{\alpha}{2}(-2\mathbf{x}(k) + 2\mathbf{w}(k)) \\
&= \mathbf{w}(k) + \alpha(\mathbf{x}(k) - \mathbf{w}(k)) \\
&= (1-\alpha)\mathbf{w}(k) + \alpha\mathbf{x}(k)
\end{aligned}$$

whereas the update rule for all other non-activated neurons is simply $\mathbf{w}(k+1) = \mathbf{w}(k)$. The update rule for the nth update in which a neuron is activated can be represented by

$$\mathbf{w}(n) = (1-\alpha)^n \mathbf{w}(0) + \alpha \sum_{k=0}^{n-1} \mathbf{x}(k)(1-\alpha)^{n-1-k} \tag{1}$$

Passing (1) to the limit as $n \to \infty$ causes the first term to vanish for $0 < \alpha < 1$, hence the weight vectors will converge to the center of the centroids for any arbitrary initial weight vector. The convergence only holds for $0 < \alpha < 1$ such that $\sum_{k=0}^{\infty} \alpha_k = \infty$. Factors that influence the convergence rate are the value of α as well as the magnitude of the initial weight vectors in comparison to the average magnitude of the data vectors. ■

Problem 4

For the following six samples from classes C_1 and C_2, use the Preceptron learning rule to arrive at the final weight vector (assume $\mathbf{w}(0) = \begin{bmatrix}1 & 1\end{bmatrix}^T$ and $b = 0$). Illustrate the process graphically.

$$\begin{aligned}
\mathcal{C}_1 &= \{\begin{bmatrix}0 & 0\end{bmatrix}^T, \begin{bmatrix}-0.5 & -1\end{bmatrix}^T, \begin{bmatrix}-1 & -2\end{bmatrix}^T\} \\
\mathcal{C}_2 &= \{\begin{bmatrix}2 & 0\end{bmatrix}^T, \begin{bmatrix}1.5 & -1\end{bmatrix}^T, \begin{bmatrix}1 & -2\end{bmatrix}^T\}
\end{aligned}$$

Solution. We assume that the preceptron algorithm should classify $\mathbf{x} \in \mathcal{C}_1$ as negative and $\mathbf{x} \in \mathcal{C}_2$ as positive. Using the preceptron learning rule of $\mathbf{w}(k+1) = \mathbf{w}(k) \pm 2\mu\mathbf{x}(k)$ we find the inner product between each ith data vector $\mathbf{x}_{1,i} \in \mathcal{C}_i$ and the current weight and update the weight vector accordingly. These quantities and the resulting updated weight vector are shown below

$$\begin{aligned}
\mathbf{x}_{1,1}^T\mathbf{w}(1) = 0 &\longrightarrow \mathbf{w}(2) = \mathbf{w}(1) \\
\mathbf{x}_{2,1}^T\mathbf{w}(2) = -1.5 &\longrightarrow \mathbf{w}(3) = \mathbf{w}(1) \\
\mathbf{x}_{3,1}^T\mathbf{w}(3) = -3 &\longrightarrow \mathbf{w}(4) = \mathbf{w}(1) \\
\mathbf{x}_{4,2}^T\mathbf{w}(4) = 2 &\longrightarrow \mathbf{w}(5) = \mathbf{w}(1) \\
\mathbf{x}_{5,2}^T\mathbf{w}(5) = 0.5 &\longrightarrow \mathbf{w}(6) = \mathbf{w}(1) \\
\mathbf{x}_{6,2}^T\mathbf{w}(6) = -1 &\longrightarrow \mathbf{w}(7) = \begin{bmatrix}-0.5 & -2\end{bmatrix}^T
\end{aligned}$$

The weight vector converges after these three iterations, correctly classifying all points. The illustration of each weight vector and all six data vectors is shown below in Figure (1). ■

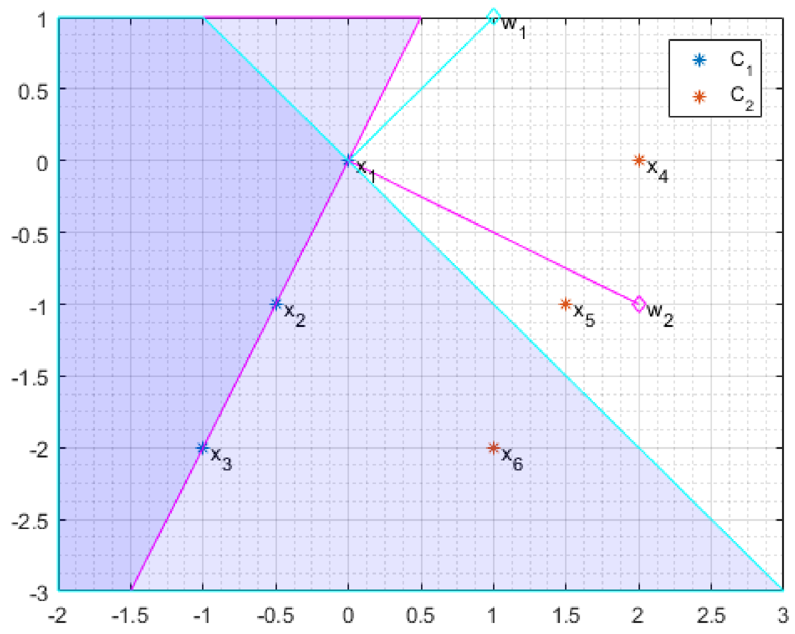

Figure 1: Weights and data vectors for preceptron algorithm