HW-2

S. Sai Ashish
EE16BTECH11043

(1) For K class linear discriminant classifier,

We consider single K-class discriminant comprising of K linear functions of the form

$$y_k(x) = w_k^T x + w_{k0}$$

and assigning a point x to class C_k if

$y_k(x) > y_j(x) \ \forall j \neq k$

The decision boundary is thus given by

$y_k(x) = y_j(x)$ given by

$$(w_k - w_j)^T x + (w_{k0} - w_{j0}) = 0$$

This decision region is always singly connected and convex (proof below)

To prove this,

consider two points x_A, x_B inside the decision region R_k.

Any point $\vec{x}$ that lies on the line connecting $\vec{x}_A$ and $\vec{x}_B$ is of the form

$$\vec{x} = \lambda \vec{x}_A + (1-\lambda)\vec{x}_B$$

where $0 \leq \lambda \leq 1$ From linearity of discriminan functions,

$$y_k(\vec{x}) = \lambda\, y_k(\vec{x}_A) + (1-\lambda)\, y_k(\vec{x}_B)$$

$$> \lambda\, y_j(\vec{x}_A) + (1-\lambda)\, y_j(\vec{x}_B)$$

$$> y_j(\vec{x})$$

$$\boxed{y_k(\vec{x}) > y_j(\vec{x})}$$

Proves that R_k is convex

Optimal Separating hyperplane

[two class supervised SVM]

assume $y \in \{-1, +1\}$

$$L(y^{(i)}, \hat{y}^{(i)}(\vec{x})) = y^{(i)}(\omega^T x^{(i)} + \omega_0) < 0$$

If we make M errors or misclassification

$$L(y, \hat{y}(\vec{x})) = \sum_{i \in M} y^{(i)} (\omega^T x^{(i)} + \omega_0)$$

Find $\vec{\omega}$ so that $L(y, \hat{y}(\vec{x}))$ is maximised

(Note :- negative sign of L)

Drawbacks : Solution depends on initial choice of $\vec{\omega}$

We can do better

$$\max_{\vec{\omega}, \omega_0, \|\vec{\omega}\|=1} u \quad ; \quad u > 0$$

Subject to the condition

$$y^{(i)} (\omega^T \cdot \vec{x}^{(i)} + \omega_0) \geq u \qquad 1 \leq i \leq N.$$

Lets remove the $||w|| = 1$ constraint

Let's reformulate,

$$\max_{\vec{w}, w_0} \vec{u} \qquad \boxed{u > 0}$$

subject to condition

$$(y^{(i)}) \frac{(w^T x_i^{(i)} + w_0)}{||w||} \geq u$$

$$\Rightarrow y^{(i)}(w^T x^{(i)} + w_0) \geq u||w|| \qquad \boxed{u > 0}$$

If $||w|| = \frac{1}{u}$ the the above optimization is ~~equivalent~~ equivalent to

$$\min_{w_0, \vec{w}} \frac{1}{2} ||w||^2$$

Such that condition)

$$y^{(i)}(w^T x^{(i)} + w_0) \geq 1$$

Using Lagrangian multipliers concept,
This converts as an unconstrained problem.

$$L_p = \min_{\vec{\omega}, \omega_0} \frac{\|\omega\|^2}{2} - \left(\sum_{i=1}^{N} \alpha_i \left(y^{(i)} (\omega^T x^i + \omega_0) - 1\right)\right)$$

Setting

$$\nabla_{\omega} L_p = 0$$

$$\boxed{\vec{\omega} = \sum_{i=1}^{N} \alpha_i y^i \vec{x}^i} \quad -(1)$$

&

$$\nabla_{\omega_0} L_p = 0$$

$$\boxed{\sum_{i=1}^{N} \alpha_i y_i = 0} \quad -(2)$$

Putting $\vec{\omega}$ in L_p & careful simplification yields the dual of L_p ie.

Maximize,

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i y^i \alpha_j y^j (x^i)^T x^j \quad -(3)$$

s.t

$$\alpha_i \geq 0 \ \forall i, \ \& \ \sum_{i=1}^{N} \alpha_i y_i = 0 \quad -(4)$$

In addition to this our optimal solutions must follow

The KKT condition

$$\boxed{\alpha_i [y^i (w^T x^i + w_0) - 1] = 0} \quad -(5)$$

Thus from the conditions (4) and (5)

ie $\alpha_i \geq 0 \ \forall i$ $\quad \alpha_i [y^{(i)} (w^T x^i + w_0) - 1] = 0$

(1) $\alpha_i \neq 0$ ie $\alpha_i > 0$

$y^{(i)} (w^T x^i + w_0) - 1 = 0$ → w_0 can be found from here corresponding to that non zero α_j

(2) $\alpha_i \neq 0$ ie. $\alpha_i > 0$

They contribute to weight vector since $\boxed{\bar{w} = \sum_{i=1}^{N} \alpha_i y^{(i)} \bar{x}^{(i)}}$

Those vectors are called support vectors.

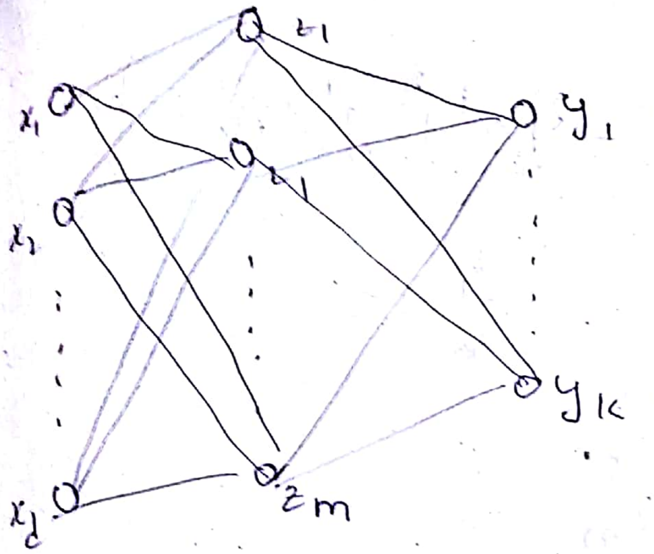
(3) Let us take the case discussed in the class

there is input layer then 'i' hidden layer and then output layer.

Let the complete set of weights be

$\{ \alpha_{0m}, \vec{\alpha}_m \ ; \ m = 1, 2, \dots M \}$ $\quad M(d+1)$ weights

$\{ \beta_{0k}, \vec{\beta}_k \ ; \ k = 1, 2, \dots K \}$ $\quad K(M+1)$ weights



Where

$$z_m = \sigma(\alpha_{0m} + \alpha_m^T x) \quad m = 1, \cdots, M$$

$$y_k = g_k(\beta_{0k} + \beta_k^T z) \quad k = 1, \cdots, K.$$

Here,

σ is Sigmoid

g is Softmax function.

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

They both are activation functions.

$$g_k(T) = \frac{e^{T_k}}{\sum_{l=1}^{K} e^{T_l}}$$

where,

$z = [z_1, z_2, \cdots z_m)$ $\quad y = (y_1, y_2, \cdots y_k)$

We use Sum of squared errors as our measure of fit

$$R(\theta) = \sum_{i=1}^{K} \sum_{i=1}^{N} \left(y_K^{(i)} - \hat{y}_K(x^i) \right)^2$$

$$= \sum_{i=1}^{N} R^{(i)}(\theta).$$

Since functions such as Sigmoid, softmax etc which have bee used are not convex we only have a optimal 'LOCAL' Minimum

The generic approach to minimizing $R(\theta)$ is by gradient descent called back propagation

Here is back propagation indetail for squared error loss function.

Let $z_m^{(i)} = \sigma(\alpha_{0m} + \alpha_m^T \vec{x}^{(i)})$,

$\vec{z}^{(i)} = (z_1^{(i)}, z_2^{(i)} \dots z_m^{(i)})$

$$R(\theta) = \sum_{i=1}^{N} R^{(i)}(\theta)$$

Scanned by CamScanner

With derivatives,

$$\frac{\partial R_i}{\partial \beta_{km}} = \underbrace{\left(-2(y_k^{(i)} - \hat{y}_k(x^{(i)}))\, g_k'(\beta_{0k}+\beta_k^T z^i)\right)}_{\delta_k^{(i)}} z_m^i$$

$$\boxed{\frac{\partial R_i}{\partial \beta_{km}} = \delta_k^{(i)} z_m^i}$$

$$\frac{\partial R_i}{\partial \alpha_{ml}} = \underbrace{\left(-\sum_{k=1}^{K} 2(y_k^{(i)} - \hat{y}_k(x^{(i)}))\, g_k'(\beta_{0k}+\beta_k^T z_i)\, \beta_{km}\, \sigma'(\alpha_{0m}+\alpha_m^T x^{(i)})\right)}_{s_{mi}} \cdot x_l^i$$

$$= s_m^{(i)} x_{il}^{(i)}$$

Given these derivatives,

A gradient descent update at the $(r+1)^{st}$ iteration

has the form

$$\beta_{km}^{(r+1)} = \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i^{(i)}}{\partial \beta_{km}^{(r)}},$$ where γ_r is the learning rate.

$$\alpha_{ml}^{(r+1)} = \alpha_{ml}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i^{(i)}}{\partial \alpha_{ml}^{(r)}}$$

At each iteration the value of weights + bias are being updated, thus this is known as back propagation since gradient of cost fn. from previous value is used to update the weights to new value,

The expression for bias is

$$\beta_{0k}^{(r+1)} = \beta_{0k}^{(r)} - \gamma_r \sum_{i=1}^{N} \delta_k^{(i)} \qquad \text{Since } \frac{\partial R^{(i)}(\theta)}{\partial \beta_{0k}} = \delta_k^{(i)}$$

$$\alpha_{0m}^{(r+1)} = \alpha_{0m}^{(r)} - \gamma_r \sum_{i=1}^{N} s_m^{(i)} \qquad \text{Since } \frac{\partial R^{(i)}(\theta)}{\partial \alpha_{0m}} = s_m^{(i)}$$

this is similar to Newton raphson method of finding $f(x)=0$ here our $f(x)$ is $\nabla R(\theta)$.

By back propagating with some learning rate.

<u>Learning Rate</u> :- Converging rate i.e by what fraction of gradient we are correcting error and updating the weights (or) bias.

(5)

For the above qn,

If the remaining set up being same as in qn 4.

$$R(\theta) = -\sum_{i=1}^{N}\sum_{k=1}^{K} y_k^{(i)} \log[\hat{y}_k(\vec{x}^i)]$$

$$= +\sum_{i=1}^{N} R^i(\theta)$$

where $R^i(\theta) = -\sum_{k=1}^{K} y_k^{(i)} \log[\hat{y}_k(\vec{x}^i)]$

with derivatives

$$\frac{\partial R^i(\theta)}{\partial \beta_{km}} = \underbrace{\left[\frac{-y_k^{(i)}}{\hat{y}_k(\vec{x}^i)} g_k'(\beta_{0k}+\beta_k^T z^i)\right]}_{\delta_k^{(i)}} z_m^i$$

$$= \delta_k^{(i)} z_m^{(i)}$$

$$\frac{\partial R^{(i)}(\theta)}{\partial \alpha_{ml}} = \left[-\sum_{k=1}^{K}\left[y_k^{(i)}/\hat{y}_k(\vec{x}^i)\right] g_k'(\beta_{0k}+\beta_k^T z^i)\beta_{km} \sigma'(\alpha_{0m}+\alpha_m^T x^{(i)})\right] x_l^i$$

$$= s_m^{(i)} x_l^{(i)}$$

A gradient descent update at the $(r+1)^{th}$ iteration has the form:

$$\beta_{km}^{(r+1)} = \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R^{(i)}(\theta)}{\partial \beta_{km}^{(r)}}$$

$$\alpha_{ml}^{(r+1)} = \alpha_{ml}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R^{(i)}\theta}{\partial \alpha_{ml}^{(r)}}$$

$$\beta_{0k}^{(r+1)} = \beta_{0k}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R^{(i)}(\theta)}{\partial \beta_{0k}^{(r)}} \quad \text{where } \frac{\partial R^{(i)}(\theta)}{\partial \beta_{0k}} = \delta_k^{(i)}$$

$$\alpha_{0m}^{(r+1)} = \alpha_{0m}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R^{(i)}\theta}{\partial \alpha_{0m}^{(r)}} \quad \text{where } \frac{\partial R^{(i)}(\theta)}{\partial \alpha_{0m}} = S_m^{(i)}$$

At each iteration the value of weights & bias are being updated from the gradient of cost function at previous value is used to update the weights to new value justifying the name back propagation.