Binary Linear Classifier

Dataset $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$, $x_i \in \mathbb{R}^d$, $y \in \{-1, +1\}$

Weight Vector $w = (w_1, w_2, \dots, w_d) \in \mathbb{R}^d$ Constant $b \in \mathbb{R}$ Given Input $x = (x_1, x_2, \dots, x_d) \in \mathbb{R}^d$

Compute $\sum_{i=1}^{d} w_i x_i > b \longrightarrow \hat{y} = +1$ $\sum_{i=1}^{d} w_i x_i < b \longrightarrow \hat{y} = -1$ Indicator Function

Model Parameters $\Omega = \{w, b\}$ y_i (True Label) $\hat{y}_i$ (Predicted Label)

Training Loss Function $E_{in}(\Omega)$ = Average # Of Miss Classified Points = $\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\{y_i \neq \hat{y}_i\}$

Optimal Model Parameters $\Omega^* = \underset{\Omega}{\operatorname{argmin}} E_{in}(\Omega)$

$x = (x_0 = 1, x_1, x_2, \dots, x_d) \in \mathbb{R}^{d+1}$

$w = (w_0 = -b, w_1, w_2, \dots, w_d) \in \mathbb{R}^{d+1}$

$w^T x = \sum_{i=0}^{d} w_i x_i = w_0 x_0 + \sum_{i=1}^{d} w_i x_i \begin{cases} > 0, \hat{y} = +1 \\ < 0, \hat{y} = -1 \end{cases}$ Decision Rule $\hat{y} = h_w(x) = \operatorname{sign}(w^T x)$

Perceptron Learning Algorithm (Learning Rate Of 1)

Input: Linearly Separable Dataset $\mathcal{D}$, Output: $w \in \mathbb{R}^{d+1}$, $E_{in}(w) = 0$

Initialize w in an arbitrary fashion, $w = (w_0 = 0, w_1 = 0, \dots, w_d = 0) \in \mathbb{R}^{d+1}$

Step 1: Check If $E_{in}(w) = 0$, If YES, Output w

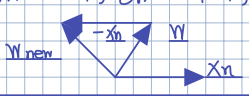
Step 2: Let (x_n, y_n) be a misclassified point in $\mathcal{D}$, If (x_n, y_n) is on the boundary, treat it as misclassified

If $y_n = +1$ then $w \longleftarrow w + x_n$ If $y_n = -1$ then $w \longleftarrow w - x_n$ Go To Step 1

Geometric Intuition

(x_n, y_n) is misclassified, $y_n = +1$, $\hat{y}_n = -1$, $w^T x_n \leq 0$

$y_n = -1$, $\hat{y}_n = +1$, $w^T x_n \geq 0$



Pocket Algorithm

Idea: At each iteration, keep aside the "best weight vector", Run PLA sufficiently many iterations

Linear Regression

Data Matrix $(N \times (d+1))$ $\mathcal{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_N^T \end{bmatrix}$ Observation Vector $(N \times 1)$ $y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$ Weight Vector $((d+1) \times 1)$ $w = \begin{bmatrix} w_0 \\ \vdots \\ w_d \end{bmatrix}$ Prediction Vector $(N \times 1)$ $\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_N \end{bmatrix}$

$\hat{y}_i = x_i^T w$

$\hat{y} = \mathcal{X} w$

$y_i = x_{i0} w_0 + x_{i1} w_1 + \dots + x_{id} w_d$

Loss Function $E_{in}(w) = \|y - \hat{y}\|^2 = \|y - \mathcal{X} w\|^2$

$w^* = \underset{w \in \mathbb{R}^{d+1}}{\operatorname{argmin}} E_{in}(w) = (\mathcal{X}^T \mathcal{X})^{-1} \mathcal{X}^T y$

Regularized Least Square

$\underset{w \in \mathbb{R}^{d+1}}{\operatorname{argmin}} \{\|y - \mathcal{X} w\|^2 + \lambda \|w\|^2\} = w_\lambda^*$ Regularization Coefficient

$w_\lambda^* = (\mathcal{X}^T \mathcal{X} + \lambda)^{-1} \mathcal{X}^T y$

Least Square Solution

Polynomical Fitting, $x \longrightarrow z = (1, x, x^2, \dots, x^d)$

$\nabla f(w) = \begin{bmatrix} \frac{\partial f(w)}{\partial w_1} & \dots & \frac{\partial f(w)}{\partial w_d} \end{bmatrix}^T \in \mathbb{R}^d$

$\nabla_w (w^T A v) = Av$ $\nabla_w (v^T A w) = (v^T A)^T = A^T v$ $\nabla_w (w^T A w) = (A + A^T) w$

Logistic Regression

$P_f(y = 1 | x) = \theta(s) = \theta(w^T x)$

Sigmoid Activation Function $\theta(s) = \frac{e^s}{1 + e^s}$ (Non Linear)

$\hat{P}_w(1|x) = \frac{e^{w^T x}}{1 + e^{w^T x}}$ $\hat{P}_w(-1|x) = \frac{e^{-w^T x}}{1 + e^{-w^T x}}$ $\hat{P}_w(y|x) = \frac{e^{y w^T x}}{1 + e^{y w^T x}}$

Input $\mathcal{D}$, Model Parameter $w \in \mathbb{R}^{d+1}$, Output $[\hat{P}_w(1|x), \hat{P}_w(-1|x)]$

Log Loss Function, $-\log \hat{P}_w(y|x)$, $e_n(w) = -\log \hat{P}_w(y_n|x_n) = \log(1 + e^{-y_n w^T x_n})$

$E_{in}(w) = \frac{1}{N}\sum_{n=1}^{N} e_n(w) = \frac{1}{N}\sum_{n=1}^{N} \log(1 + e^{-y_n w^T x_n})$ $w^* = \underset{w \in \mathbb{R}^{d+1}}{\operatorname{argmin}} E_{in}(w)$ $w_\lambda^* = \underset{w \in \mathbb{R}^{d+1}}{\operatorname{argmin}} (E_{in}(w) + \lambda \|w\|^2)$

Maximum Likelihood Viewpoint

$P_r(y_1, y_2 \dots y_N | x_1, x_2 \dots x_N) = \prod P_r(y_n | x_n)$ Independent Event

$w^* = \underset{w \in \mathbb{R}^{d+1}}{\operatorname{argmax}} \prod_{n=1}^{N} P_r(y_n|x_n) = \underset{w \in \mathbb{R}^{d+1}}{\operatorname{argmax}} \log\left(\prod_{n=1}^{N} P_r(y_n|x_n)\right) = \underset{w \in \mathbb{R}^{d+1}}{\operatorname{argmax}} \sum_{n=1}^{N} \log(P_r(y_n|x_n)) = \underset{w \in \mathbb{R}^{d+1}}{\operatorname{argmin}} E_{in}(w)$

Cross Entropy Viewpoint

$S = \{s_1, s_2, \dots, s_M\}$, $P = (p(s_1), p(s_2), \dots, p(s_M))$, $Q = (q(s_1), q(s_2), \dots, q(s_M))$

$CE(P, Q) = -\sum_{i=1}^{M} p(s_i) \log q(s_i)$ $e_n(w) = -\{\mathbb{1}\{y_n = +1\} \log \hat{P}_w(1|x_n) + \mathbb{1}\{y_n = -1\} \log \hat{P}_w(-1|x_n)\}$

$P_n = (\mathbb{1}\{y_n = +1\}, \mathbb{1}\{y_n = -1\})$ $Q_n = (\log \hat{P}_w(1|x_n), \log \hat{P}_w(-1|x_n))$ $E_{in}(w) = \frac{1}{N}\sum_{n=1}^{N} CE(P_n, \hat{P}_w(x_n))$

Knowledge Distillation

Transfer knowledge from a large model to a smaller one

Training For Logistic Regression (No Analytical Solution) (Gradient Descent)

If $x < x^*$, $f(x)$ is decreasing, $f'(x) < 0$

$x > x^*$, $f(x)$ is increasing, $f'(x) > 0$

1. Initialize $x = x_0$
2. If $f'(x) \approx 0$, stop and output x
3. If $f'(x) > 0$, $x = x - \varepsilon$
4. If $f'(x) < 0$, $x = x + \varepsilon$
5. Go to step 2

ε (Step Size) (eg 10^{-3})

$x_1 = x_0 + \delta u \quad ||u|| = 1 \quad u^* = \arg\min_{u} f(x_0 + \delta u)$ Taylor Approximation $f(x_0 + \delta u) \approx f(x_0) + \delta u^T \nabla f(x_0)$

$u^* = \frac{-\nabla f(x_0)}{||\nabla f(x_0)||}, \; x_t = x_{t-1} + \delta_t u^* \quad u \in \mathbb{R}^2$ $\quad x_t = x_{t-1} - \varepsilon_t \nabla f(x_{t-1})$ (Step Size)$_t \propto \varepsilon_t ||\nabla f(x_{t-1})||$

Gradient Descent Algorithm Initialize x_0 in some arbitrary fashion, $t = 0, 1, 2, \ldots$, Compute $g_t = \nabla f(x_t)$ Select Direction $u_t = -g_t$, Update $x_{t+1} = x_t + \varepsilon_t u_t$, Continue until $||\nabla f(x_t)|| \approx 0$

Linear Regression w_t will converge to w_{LS}^* $\quad \nabla E_{in}(w_t) = \frac{1}{N}\sum_{n=1}^{N} \nabla e_n(w)$ $\quad \nabla_w e_n(w) = 2(w^T x_n - y_n)x_n$ $\quad w_{t+1} = w_t - \varepsilon_t \frac{2}{N}\sum_{n=1}^{N}(w^T x_n - y_n)x_n$ $O(Nd)$

Stochastic Gradient Descent $w_{t+1} = w_t - \varepsilon_t \nabla e_n(w_t) \quad n \in (\text{Uniform}\{1, 2, \ldots, N\}$

Mini Batch Gradient Descent Draw M examples at random without replacement $w_{t+1} = w_t - \frac{\varepsilon_t}{M}\sum_{n=1}^{M} \nabla e_{(n)}(w_t)$

SGD Update Rule $w_{k+1} = w_k + \varepsilon_k \frac{y_n x_n}{1 + e^{y_n w_k^T x_n}}$ (Logistic Regression)

Multiple Classes Logistic Regression $\Omega = \{w(1), w(2), w(3)\}$ $\quad [\hat{P}_\Omega(1|x), \hat{P}_\Omega(2|x), \hat{P}_\Omega(3|x)]$

$\hat{P}_\Omega(i|x) = \frac{e^{w(i)^T x}}{e^{w(1)^T x} + e^{w(2)^T x} + e^{w(3)^T x}} \quad e_n(\Omega) = -\log \hat{P}_\Omega(y_n|x_n) \quad w(1) - w(2) = w$

$\Omega_{k+1} = \Omega_k - \varepsilon_k \nabla_{\Omega_k} e_n(\Omega_k)$

SGD + Momentum Velocity Vector $v_0 = 0$, $g_t = \nabla e_n(w_t)$, $v_t = -\varepsilon_t g_t + \mu v_{t-1}$, $w_{t+1} = w_t + v_t$ (μ: Momentum Coefficient)

Nestor Momentum $v_t = \mu v_{t-1} - \varepsilon_t \nabla e_n(w_t + \mu v_{t-1})$, $w_{t+1} = w_t + v_t$ $\quad$ GD: $O(1/k)$, Nestor: $O(1/k^2)$

Ada Grad $g_t = \nabla e_n(w_t) \in \mathbb{R}^d$, $\gamma_t = \gamma_{t-1} + g_t \odot g_t$, $\gamma_t(j) = \sum_{q=1}^{t} g_q^2(j)$ $\quad w_{t+1}(j) = w_t(j) - \frac{\varepsilon_t}{\sqrt{\gamma_t(j)}} g_t(j)$

RMS — PROP $\gamma_t = \beta \gamma_{t-1} + (1-\beta) g(t) \odot g(t)$ $\quad w_{t+1} = w_t - \frac{\varepsilon_t}{\sqrt{\gamma_t}} \odot g(t)$

ADAM $v_t = \mu v_{t-1} - \frac{\varepsilon_t}{\sqrt{\gamma_t}} \odot g(t)$ $\quad w_{t+1} = w_t + v_t$

For Classification, $\emptyset \longrightarrow$ Sigmoid, For Regression, $\emptyset \longrightarrow$ Identity Function $\quad \hat{y}_1 = OR(h_1, h_2)$ $\hat{y}_2 = AND(h_1, h_2)$

$OR(h_1, h_2) = \text{sign}(h_1 + h_2 + 1.5)$ $\quad AND(h_1, h_2) = \text{sign}(h_1 + h_2 - 1.5)$ $\quad XOR(h_1, h_2) = AND(\hat{y}_1, \hat{y}_2)$ $\quad (M = 0.5)$

Layer 1, Implement l_1, l_2, l_3 to output h_1, h_2, h_3 $\quad$ Layer 2, Implement y_1, y_2, y_3 $\quad$ Layer 3, $y = OR(y_1, y_2, y_3)$

Neural Network Input Layer $l = 0$, Hidden Layers $1 \le l \le L-1$, Output Layer L, $w_{i,j}^{(l)}$ weight between node i in $l-1$ and j in l

$d^{(l)}$ Number of nodes excluding bias node in layer l, $x_j^{(l)}$ Output from node j in layer l, $s_j^{(l)}$ Input to node j in layer l

$$s^{(l)} = \begin{bmatrix} s_1^{(l)} \\ \vdots \\ s_{d^{(l)}}^{(l)} \end{bmatrix} \quad \theta(s^{(l)}) = \begin{bmatrix} \theta(s_1^{(l)}) \\ \vdots \\ \theta(s_{d^{(l)}}^{(l)}) \end{bmatrix} \quad x^{(l)} = \begin{bmatrix} 1 \\ \vdots \\ s^{(l)} \end{bmatrix} \quad w^{(l)} = \left\{ w_{i,j}^{(l)} \right\} \begin{matrix} \text{Row } 0 \le i \le d^{(l-1)} \\ \text{Column } 1 \le j \le d^{(l)} \end{matrix}$$

Forward Propagation Algorithm

Input $x^{(0)} = [1 \; x_1 \; x_2 \; \ldots \; x_d]^T$

for $l = 1, 2, \ldots, L$ do

$s^{(l)} = w^{(l)T} x^{(l-1)}$

$x^{(l)} = \theta(s^{(l)})$

$\sum_{l=1}^{L}(d^{(l-1)} + 1)d^{(l)} + \sum_{l=1}^{L} d^{(l)}$

Output $x^{(L)}$

Backward Propagation Algorithm Backward Message

Input (x, y), $\Omega = \{w^{(1)}, \ldots, w^{(L)}\}$ Output $\delta^{(l)}$ for $l = 1, 2, \ldots, L$

$\frac{\partial e}{\partial w_{i,j}^{(l)}} \; \forall i, j, l$ 1. Run forward propagation to compute $\{s^{(l)}, x^{(l)}\}$ and $e(\Omega) = g(x^{(L)}, y)$

2. $\delta^{(L)} = \nabla_{x^{(L)}} g(x^{(L)}, y) \odot \theta'(s^{(L)})$

3. For $l = L-1, \ldots, 1$

$\delta^{(l)} = [w^{(l+1)} \delta^{(l+1)}]_1^{d^{(l)}} \odot \theta'(s^{(l)})$

$\frac{\partial e}{\partial w_{i,j}^{(l)}} = x_i^{(l-1)} \delta_j^{(l)}$ $\quad w_{i,j}^{(l)} \leftarrow w_{i,j}^{(l)} - \varepsilon_k \frac{\partial e}{\partial w_{i,j}^{(l)}}$

Linear Regression Minimizer Weights

$E_{in}(w) = \frac{1}{N}||y - Xw||^2 = \frac{1}{N}(y - Xw)^T(y - Xw)$

$= \frac{1}{N}[y^T y - y^T X w - w^T X^T y + w^T X^T X w]$

$\frac{\partial E_{in}(w)}{\partial w} = \frac{1}{N}[-X^T y - X^T y + (X^T X + X^T X)w]$

$0 = -\frac{1}{N}[2X^T X w - 2X^T y] \quad X^T X w = X^T y \quad w = (X^T X)^{-1} X^T y$

Matrix Inverse $\text{Adj}(A) = \text{Cofactor}(A)^T$

$\frac{1}{\det(A)} \text{Adj}(A)$

$\text{Sign}(x) = \begin{cases} -1, & x \le 0 \\ 1, & x > 0 \end{cases}$

Sigmoid $\theta(s) = \frac{e^s}{1 + e^s}$, $\theta'(s) = \theta(s)(1 - \theta(s))$

$\tanh(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}}$ $\quad \theta'(s) = (1 - \tanh^2(s))$

ReLU $\theta(s) = \max(0, s)$

Leaky ReLU $\theta(s) = \begin{cases} s, & s \ge 0 \\ \alpha s, & s < 0 \end{cases}$

Data Augmentation

Weights Initialization