Valore atteso $E[X] = \sum_{i=1}^{N} (x_i \cdot P(X = x_i))$

varianza $Var(X) = E[(X-\mu)^2]$; μ = media di X

Simple linear regression: valore reale $y_i = w_0 + w_1 x_i + \varepsilon_i$

valore previsto $\hat{y}_i = f(x_i) = w_0 + w_1 x_i$

errore da minimiz. $RSS(w_0, w_1) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{N} [y_i - (w_0 + w_1 x_i)]^2$

$$\nabla RSS(w_0, w_1) = \begin{bmatrix} \frac{\partial RSS}{\partial w_0} \\ \frac{\partial RSS}{\partial w_1} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} 2 [y_i - (w_0 + w_1 x_i)] \cdot (-1) \\ \sum_{i=1}^{N} 2 [y_i - (w_0 + w_1 x_i)] \cdot (-x_i) \end{bmatrix}$$

$$= \begin{bmatrix} -2 \sum_{i=1}^{N} [y_i - (w_0 + w_1 x_i)] \\ -2 \sum_{i=1}^{N} [y_i - (w_0 + w_1 x_i)] \cdot (x_i) \end{bmatrix}$$

Forma chiusa

$$\begin{cases} -2 \Sigma [y_i - (w_0 + w_1 x_i)] = 0 \\ -2 \Sigma [y_i - (w_0 + w_1 x_i)](x_i) = 0 \end{cases}$$

$$\begin{cases} \Sigma y_i - \hat{w}_0 \sum_{i=1}^{N} 1 - \hat{w}_1 \Sigma x_i = 0 \\ \Sigma x_i y_i - \hat{w}_0 \Sigma x_i - \hat{w}_1 \Sigma x_i^2 = 0 \end{cases}$$

$$\begin{cases} \hat{w}_0 = \frac{\Sigma y_i}{N} - \hat{w}_1 \frac{\Sigma x_i}{N} \\ \hat{w}_1 = \frac{\Sigma x_i y_i - \hat{w}_0 \Sigma x_i}{\Sigma x_i^2} = \frac{\Sigma x_i y_i - \frac{\Sigma y_i \Sigma x_i}{N}}{\Sigma x_i^2 - \frac{(\Sigma x_i)^2}{N}} \end{cases}$$

Gradient Descent

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \alpha \nabla RSS(\mathbf{w}^{(t)}), \quad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

$$\begin{cases} w_0^{(t+1)} \leftarrow w_0^{(t)} - \alpha \frac{\partial RSS(\mathbf{w}^{(t)})}{\partial w_0} \\ w_1^{(t+1)} \leftarrow w_1^{(t)} - \alpha \frac{\partial RSS(\mathbf{w}^{(t)})}{\partial w_1} \end{cases}$$

algoritmo:

$w^{(t)} = 0$

$t = 1$

while $\|\nabla RSS(w^{(t)})\|_2 > \varepsilon$

$$w_0^{(t+1)} \leftarrow w_0^{(t)} + 2\alpha \Sigma [y_i - \hat{y}_i(\mathbf{w}^{(t)})]$$

$$w_1^{(t+1)} \leftarrow w_1^{(t)} + 2\alpha \Sigma [y_i - \hat{y}_i(\mathbf{w}^{(t)})] x_i$$

$t \leftarrow t+1$

Multiple Features

Gradient Descent con notazione matriciale

$$y_i = [\phi_0(x_i) \dots \phi_D(x_i)] \begin{bmatrix} w_0 \\ \dots \\ w_D \end{bmatrix} + \varepsilon_i = \phi^T(x_i) \mathbf{w} + \varepsilon_i$$

algoritmo

$w^{(t)} = 0$

$t = 1$

while $\|\nabla RSS(w^{(t)})\|_2 > \varepsilon$

for $j = 0, 1, \dots, D$

derivata_parziale[j] $= -2 \sum_{i=1}^{N} \phi_j(x_j) [y_i - \hat{y}_i(w^{(t)})]$

$w_j^{(t+1)} \leftarrow w_j^{(t)} - \alpha *$ derivata_parziale[j]

$t \leftarrow t+1$

Loss Function: $\mathcal{L}[y, f_{\hat{w}}(x)] = \mathcal{L}[y, \hat{y}]$ se è squared error: $\mathcal{L}[y, f_{\hat{w}}(x)] = [y - f_{\hat{w}}(x)]^2$

Training error: $\text{TRAINING } E = \frac{1}{N_{TR}} \cdot \sum_{i=1}^{N} \mathcal{L}[y_i, f_{\hat{w}}(x_i)]$

Generalization/True error: $\text{GEN } E = E_{x,y}[\mathcal{L}(y, f_{\hat{w}}(x))]$

Test error: $\text{TEST } E = \frac{1}{N_{TEST}} \sum_{i \in TEST} \mathcal{L}[y_i, f_{\hat{w}}(x_i)]$

Bias: $\text{BIAS}(f_{\hat{w}}(x_t)) = f_{w(TRUE)}(x_t) - f_{\bar{w}}(x_t)$

$f_{\bar{w}}(x_t) = E_{train}[f_{\hat{w}}(x_t)]$

variance: $\text{Var}(f_{\hat{w}}(x_t)) = E_{train}[(f_{\hat{w}}(x_t) - f_{\bar{w}}(x_t))^2]$

Ridge Regression: costo_ridge = fitness + misura di grandezza dei coeff.

$\text{costo\_ridge\_L2norm}(w) = RSS(w) + \lambda \|w\|_2^2 = (y - \Phi w)^T(y - \Phi w) + \lambda w^T w$

↳ se $\lambda = 0$: min RSS → $\hat{w}_{LS}$

↳ se $\lambda \to \infty$: $\forall$ soluz con $\hat{w} \neq 0$, costo → $+\infty$. unica soluz $\hat{w}_{RIDGE} = 0$

↳ se $0 < \lambda < \infty$: $0 \leq \|\hat{w}_{RIDGE}\|_2^2 \leq \|\hat{w}_{LS}\|_2^2$

$$\nabla \text{costo\_ridge\_L2norm}(w) = \nabla[(y - \Phi w)^T(y - \Phi w) + \lambda w^T w]$$
$$= \nabla[(y - \Phi w)^T(y - \Phi w)] + \nabla[\lambda w^T w]$$
$$= -2\Phi^T(y - \Phi w) + \lambda 2w$$

Forma chiusa
$$-2\Phi^T(y - \Phi w) + \lambda 2w = 0$$
$$-2\Phi^T y + 2\Phi^T \Phi \hat{w} + 2\lambda \hat{w} = 0$$
$$\Phi^T \Phi \hat{w} + \lambda I \hat{w} = \Phi^T y$$
$$(\Phi^T \Phi + \lambda I)\hat{w} = \Phi^T y$$
$$(\Phi^T \Phi + \lambda I)^{-1}(\Phi^T \Phi + \lambda I)\hat{w} = (\Phi^T \Phi + \lambda I)^{-1}\Phi^T y$$
$$Iw = (\Phi^T \Phi + \lambda I)^{-1}\Phi^T y = \hat{w}_{Ridge}$$

$$\nabla \text{costo\_ridge\_L2norm}(w) = \begin{bmatrix} \frac{\partial \text{costo\_ridge\_L2norm}(w)}{\partial w_0} \\ \dots \\ \frac{\partial \text{costo\_ridge\_L2norm}(w)}{\partial w_D} \end{bmatrix}$$

siccome $\text{costo\_ridge}(w) = RSS(w) + \lambda \|w\|_2^2 = RSS(w) + \lambda w^T w$

$$\frac{\partial \text{costo\_ridge\_L2norm}(w)}{\partial w_j} = \frac{\partial RSS(w)}{\partial w_j} + \lambda \frac{\partial (w^T w)}{\partial w_j}$$
$$= -2 \sum_{i=1}^{N} \phi_j(x_i)[y_i - \hat{y}_i(w)] + \lambda 2 w_j$$

Gradient Descent

$w^{(t+1)} \leftarrow w^{(t)} - \alpha \nabla \text{costo\_ridge\_L2norm}(w^{(t)})$

per il generico peso

$w_j^{(t+1)} \leftarrow w_j^{(t)} - \alpha \left\{-2 \sum_{i=1}^{N} \phi_j(x_i)[y_i - \hat{y}_i(w^{(t)})] + \lambda 2 w_j^{(t)}\right\}$ cioè

$w_j^{(t+1)} \leftarrow (1 - 2\alpha\lambda) w_j^{(t)} - \alpha \left\{-2 \sum_{i=1}^{N} \phi_j(x_i)[y_i - \hat{y}_i(w^{(t)})\right\}$

algoritmo

$w^{(t)} = 0$

$t = 1$

while $\|\nabla \text{costo\_ridge\_L2norm}(w^{(t)})\|_2 > \varepsilon$

  for $j = 0, 1, \dots, D$

    $\text{der\_parz\_RSS}[j] = -2 \sum_{i=1}^{N} \phi_j(x_i)[y_i - \hat{y}_i(w^{(t)})]$

    $w_j^{(t+1)} \leftarrow (1 - 2\alpha\lambda) w_j^{(t)} - \alpha * \text{der\_parz\_RSS}[j]$

  $t \leftarrow t + 1$

k- fold cross validation, algoritmo:
∀ scelta di λ:
for $k = 1, \dots, K$
stima di $\hat{w}_\lambda$ sui blocchi di training
calcolo dell'errore sul validation block : $error_k(\lambda)$
calcolo dell'average error $CV(\lambda) = \frac{1}{K} \sum_{k=1}^{K} error_k(\lambda)$
scelta di λ^* che minimizza $CV(\lambda)$

LASSO: Least Absolute Shrinkage and Selection operator
$COSTO\_lasso(w) = RSS(w) + \lambda \|w\|_1 = \sum_{i=1}^{N} \left[y_i - \sum_{j=0}^{D} w_j \Phi_j(x_i) \right]^2 + \lambda \sum_{j=0}^{D} |w_j|$
↳ se $\lambda = 0$: min RSS → $\hat{w}_{LS}$
↳ se $\lambda \to \infty$: ∀ soluz con $\hat{w} \neq 0$, costo → $+\infty$. unica soluz $\hat{w}_{LASSO} = 0$
↳ se $0 < \lambda < \infty$: $0 \leq \| \hat{w}_{LASSO} \|_1 \leq \| \hat{w}_{LS} \|_1$
derivazione di $COSTO\_lasso(w)$ e subgradiente
↳ derivazione del termine RSS

$$\frac{\partial RSS}{\partial w_j} = -2 \sum_{i=1}^{N} \Phi_j(x_i) [y_i - \hat{y}_i(w)] = -2 \sum_{i=1}^{N} \Phi_j(x_i) \left[y_i - \sum_{j=0}^{D} w_j \Phi_j(x_i) \right] =$$

$$= -2 \sum_{i=1}^{N} \Phi_j(x_i) \left[y_i - \sum_{k \neq j} w_k \Phi_k(x_i) - w_j \Phi_j(x_i) \right]$$

$$= -2 \underbrace{\sum_{i=1}^{N} \Phi_j(x_i) \left[y_i - \underbrace{\sum_{k \neq j} w_k \Phi_k(x_i)}_{\text{predizione senza } \Phi_j} \right]}_{\rho_j} + 2 w_j \underbrace{\sum_{i=1}^{N} \Phi_j^2(x_i)}_{z_j}$$

$$= -2\rho_j + 2 w_j z_j$$

↳ derivazione del termine L1 Penalty
$\lambda \cdot \frac{\partial |w_j|}{\partial w_j}$: la presenza del modulo crea problemi quindi si usa il subgradiente

il subgradiente di g in v è un vettore s tale che $g(w) \geq g(v) + s^T (w - v)$
l'insieme dei subgradienti di g in v è detto Differential Set $\partial g(v)$
il subgradiente del modulo nel punto non derivabile varia da -1 a +1
il Differential Set è: $\partial_{w_j} |w_j| = \begin{cases} \{-1\} & \text{se } w_j < 0 \\ [-1, +1] & \text{se } w_j = 0 \\ \{1\} & \text{se } w_j > 0 \end{cases}$
e per $\lambda \|w\|_1$: $\partial \lambda |w_j| = \begin{cases} -\lambda & \text{se } w_j < 0 \\ [-\lambda, +\lambda] & \text{se } w_j = 0 \\ \lambda & \text{se } w_j > 0 \end{cases}$
↳ "derivazione" complessiva:

$$\partial_{w_j} [COSTO\_LASSO] = -2\rho_j + 2 w_j z_j + \begin{cases} -\lambda & \text{se } w_j < 0 \\ [-\lambda, +\lambda] & \text{se } w_j = 0 \\ \lambda & \text{se } w_j > 0 \end{cases} = \begin{cases} 2 w_j z_j - 2\rho_j - \lambda & \text{se } w_j < 0 \\ [-2\rho_j - \lambda, -2\rho_j + \lambda] & \text{se } w_j = 0 \\ 2 w_j z_j - 2\rho_j + \lambda & \text{se } w_j > 0 \end{cases}$$

Forma chiusa
↳ se $w_j < 0$: $2 w_j z_j - 2\rho_j - \lambda = 0 \to \hat{w}_j = \frac{2\rho_j + \lambda}{2 z_j} = \frac{\rho_j + \frac{\lambda}{2}}{z_j} < 0 \to \rho_j + \frac{\lambda}{2} < 0 \to \rho_j < -\frac{\lambda}{2}$
↳ se $w_j = 0$: $0 \in [-2\rho_j - \lambda, -2\rho_j + \lambda] \to \begin{cases} -2\rho_j - \lambda \leq 0 \\ -2\rho_j + \lambda \geq 0 \end{cases} \to \begin{cases} \rho_j \geq -\frac{\lambda}{2} \\ \rho_j \leq +\frac{\lambda}{2} \end{cases} \to -\frac{\lambda}{2} \leq \rho_j \leq \frac{\lambda}{2}$
↳ se $w_j > 0$: $2 w_j z_j - 2\rho_j + \lambda = 0 \to \hat{w}_j = \frac{2\rho_j - \lambda}{2 z_j} = \frac{\rho_j - \frac{\lambda}{2}}{z_j} < 0 \to \rho_j - \frac{\lambda}{2} < 0 \to \rho_j < \frac{\lambda}{2}$

cioè $w_j = \begin{cases} \frac{\rho_j + \frac{\lambda}{2}}{z_j} & \text{se } \rho_j < -\frac{\lambda}{2} \\ 0 & \text{se } \rho_j \in [-\frac{\lambda}{2}, \frac{\lambda}{2}] \\ \frac{\rho_j - \frac{\lambda}{2}}{z_j} & \text{se } \rho_j > \frac{\lambda}{2} \end{cases}$

Gradient Descent

Normalizzazione feature generica: $\underline{\Phi}_j(x_k) = \frac{\Phi_j(x_k)}{\sqrt{\sum_{i=1}^{N} \Phi_j^2(x_i)}} = \frac{\Phi_j(x_k)}{\sqrt{z_j}} = \frac{\Phi_j(x_k)}{Z_j}$

algoritmo con feature normalizzare

$\hat{\omega} = 0$

while not converged

for $j = 0, 1, \dots, D$

calcola $\rho_j = \sum_{i=1}^{N} \underline{\Phi}_j(x_i) \left[y_i - \hat{y}_i(\hat{\omega}_{-j}) \right]$

set $\hat{\omega}_j = \begin{cases} \rho_j + \frac{\lambda}{2} & \text{se } \rho_j < -\frac{\lambda}{2} \\ 0 & \text{se } \rho_j \in \left[-\frac{\lambda}{2}, \frac{\lambda}{2} \right] \\ \rho_j - \frac{\lambda}{2} & \text{se } \rho_j > \frac{\lambda}{2} \end{cases}$

algoritmo senza feature normalizzare

calcola $z_j = \sum_{i=1}^{N} \Phi_j(x_i)^2$

$\hat{\omega} = 0$

while not converged

for $j = 0, 1, \dots, D$

calcola $\rho_j = \sum_{i=1}^{N} \Phi_j(x_i) \left[y_i - \hat{y}_i(\hat{\omega}_{-j}) \right]$

set $\hat{\omega}_j = \begin{cases} \frac{\rho_j + \frac{\lambda}{2}}{z_j} & \text{se } \rho_j < -\frac{\lambda}{2} \\ 0 & \text{se } \rho_j \in \left[-\frac{\lambda}{2}, \frac{\lambda}{2} \right] \\ \frac{\rho_j - \frac{\lambda}{2}}{z_j} & \text{se } \rho_j > \frac{\lambda}{2} \end{cases}$

Expected Prediction Error (EPE): dà una valutazione delle prestazioni del sistema rispetto alla scelta di training set. Per ogni possibile scelta di training set se ne valuta il Generalization error. Poi si calcola l'Expected value del generalization error sui possibili training set pesati con la loro probabilità $EPE = E_{train}[GE \text{ per } \hat{w}(train)]$

Dato un training set, una $f_{\hat{w}}$ (training set), un punto x_t e $\mathcal{L}[y, f_{\hat{w}}(x)] = [y - f_{\hat{w}}(x)]^2$

si dimostra $EPE(x_t) = \sigma^2 + [bias(f_{\hat{w}}(x_t))]^2 + var[f_{\hat{w}}(x_t)]$

$EPE = E_{train}[GE \text{ per } \hat{w}(train)] = E_{train}[E_{x,y}[\mathcal{L}(y, f_{\hat{w}(train)}(x))]]$ diventa

$EPE(x_t) = E_{train, y_t}[(y_t - \hat{y}_t)^2] = E_{train, y_t}[(y_t - f_{\hat{w}(train)}(x_t))^2]$

$= E_{train, y_t}[(y_t + f_{w(true)}(x_t) + f_{w(true)}(x_t) - f_{\hat{w}(train)}(x_t))^2] = E_{train, y_t}[((y_t - f) + (f - \hat{f}))^2]$

$= E_{train, y_t}[(y_t - f)^2 + 2(y_t - f)(f - \hat{f}) + (f - \hat{f})^2]$

$= \underbrace{E_{train, y_t}[(y_t - f)^2]}_{①} + \underbrace{2\, E_{train, y_t}[(y_t - f)(f - \hat{f})]}_{②} + \underbrace{E_{train, y_t}[(f - \hat{f})^2]}_{③}$

① $E_{train, y_t}[(y_t - f)^2] = E_{y_t}[(y_t - f)^2] = E_{y_t}[\varepsilon^2] = \sigma^2$

② $2\, E_{train, y_t}[(y_t - f)(f - \hat{f})] = 2 \cdot E_{train, y_t}[(\varepsilon) \cdot (f - \hat{f})] = 2 \cdot E_{y_t}[\varepsilon] \cdot E_{train}[(f - \hat{f})] = 2 \cdot 0 \cdot \dots = 0$

③ $E_{train, y_t}[(f - \hat{f})^2] = E_{train}[(f - \hat{f})^2] = MSE(\hat{f})$ (Means Squared Error)

Dimostrazione $MSE(\hat{f}) = [bias(\hat{f})]^2 + var(\hat{f})$

$MSE[f_{\hat{w}(train)}(x_t)] = E_{train}[(f_{w(true)}(x_t) - f_{\hat{w}(train)}(x_t))^2] = E_{train}[(f_{w(true)}(x_t) - f_{\bar{w}}(x_t) + f_{\bar{w}}(x_t) - f_{\hat{w}(train)}(x_t))^2] =$

$E_{train}[((f - \bar{f}) + (\bar{f} - \hat{f}))^2] = E_{train}[(f - \bar{f})^2 + 2(f - \bar{f})(\bar{f} - \hat{f}) + (\bar{f} - \hat{f})^2] = \underbrace{E_{train}[(f - \bar{f})^2]}_{①} + \underbrace{2\, E_{train}[(f - \bar{f})(\bar{f} - \hat{f})]}_{②} + \underbrace{E_{train}[(\bar{f} - \hat{f})^2]}_{③}$

① Dato che $\bar{f} = E_{train}[\hat{f}]$: $E_{train}[(f - \bar{f})^2] = (f - \bar{f})^2 = [bias(\hat{f})]^2$

② $2\, E_{train}[(f - \bar{f})(\bar{f} - \hat{f})] = 2 \cdot (f - \bar{f}) \cdot E_{train}[(\bar{f} - \hat{f})] = 2(f - \bar{f})(\bar{f} - E_{train}[\hat{f}]) = 2(\bar{f} - \hat{f})(\bar{f} - \bar{f}) = 2(\bar{f} - \hat{f}) \cdot 0 = 0$

③ $E_{train}[(\bar{f} - \hat{f})^2] = E_{train}[(\hat{f} - \bar{f})^2] = var(\hat{f})$

$EPE(x_t) = \sigma^2 + MSE(f_{\hat{w}}(x_t)) = \sigma^2 + [bias(f_{\hat{w}}(x_t))]^2 + var[f_{\hat{w}}(x_t)]$

Classification

Classificazione Binaria: $\text{score}(x_i) = \sum_{j=0}^{D} w_j \phi_j(x_i) = \mathbf{w}^T \boldsymbol{\phi}(x_i)$

$\hat{y}_i = \text{sign}[\text{score}(x_i)] = \begin{cases} +1 & \text{se score} > 0 \\ -1 & \text{se score} < 0 \end{cases}$

$P(y_i = +1 \mid x_i, w) = g[\text{score}(x_i, w)]$ con g link function

g Logistica / Sigmoide: $\text{sigmoid}(\text{score}) = \frac{1}{1 + e^{-\text{score}}}$

Logistic Regression Model: $\hat{P}(y_i = +1 \mid x_i, \hat{w}) = \frac{1}{1 + e^{-\hat{w}^T \phi(x_i)}}$

Maximum Likelihood Estimation

$\mathcal{L}(w) = P(y_1 \mid x_1, w) \cdot \ldots \cdot P(y_N \mid x_N, w) = \prod_{i=1}^{N} P(y_i \mid x_i, w)$ che voglio massimizzare: $\max_w \mathcal{L}(w)$

per facilitare il calcolo della derivata considero il Log-Likelihood

$\ln \mathcal{L}(w) = \ln \prod_{i=1}^{N} P(y_i \mid x_i, w) = \sum_{i=1}^{N} \ln P(y_i \mid x_i, w)$

e considero separatamente i casi positivi e negativi con $I[\text{event}] = \begin{cases} 1 \text{ if event is true} \\ 0 \text{ if event is !true} \end{cases}$

$$\ln \mathcal{L}(w) = \sum_{i=1}^{N} \left\{ I[y_i = +1] \ln P(y_i = +1 \mid x_i, w) + I[y_i = -1] \ln P(y_i = -1 \mid x_i, w) \right\}$$

sostituisco le probabilità: ① $P(y_i = +1 \mid x_i, w) = \frac{1}{1 + e^{-w^T \phi(x_i)}}$

② $P(y_i = -1 \mid x_i, w) = 1 - P(y_i = +1 \mid x_i, w) = 1 - \frac{1}{1 + e^{-w^T \phi(x_i)}} = \frac{e^{-w^T \phi(x_i)}}{1 + e^{-w^T \phi(x_i)}}$

Per un solo punto x_i:

$$\begin{aligned} \ln \mathcal{L}(w) &= I[y_i = +1] \ln P(y_i = +1 \mid x_i, w) + I[y_i = -1] \ln P(y_i = -1 \mid x_i, w) \\ &= I[y_i = +1] \ln \frac{1}{1 + e^{-w^T \phi(x_i)}} + (1 - I[y_i = +1]) \ln \frac{e^{-w^T \phi(x_i)}}{1 + e^{-w^T \phi(x_i)}} \\ &= -I[y_i = +1] \ln (1 + e^{-w^T \phi(x_i)}) + (1 - I[y_i = +1]) \left[-w^T \phi(x_i) - \ln (1 - e^{-w^T \phi(x_i)}) \right] \\ &= -(1 - I[y_i = +1]) w^T \phi(x_i) - \ln (1 + e^{-w^T \phi(x_i)}) \end{aligned}$$

Derivata Parziale per un solo punto x_i:

$$\begin{aligned} \frac{\partial \ln \mathcal{L}(w)}{\partial w_j} &= -(1 - I[y_i = +1]) \frac{\partial (w^T \phi(x_i))}{\partial w_j} - \frac{\partial}{\partial w_j} \ln (1 + e^{-w^T \phi(x_i)}) \\ &= -(1 - I[y_i = +1]) \phi_j(x_i) - \phi_j(x_i) P(y_i = -1 \mid x_i, w) \\ &= \phi_j(x_i) \{ I[y_i = +1] - P(y_i = +1 \mid x_i, w) \} \end{aligned}$$

Derivata Parziale su tutti i punti:

$$\frac{\partial \ln \mathcal{L}(w)}{\partial w_j} = \sum_{i=1}^{N} \phi_j(x_i) \{ I[y_i = +1] - P(y_i = +1 \mid x_i, w) \}$$

Gradient Ascent, algoritmo

$w^{(1)} = 0$

$t = 1$

while $\| \nabla \ln \mathcal{L}(w^{(t)}) \|_2 > \varepsilon$:

  for $j = 0, 1, \ldots, D$

    $\text{der\_parz}[j] = \sum_{i=1}^{N} \phi_j(x_i) \{ I[y_i = +1] - P(y_i = +1 \mid x_i, w^{(t)}) \}$

    $w_j^{(t+1)} \leftarrow w_j^{(t)} + \alpha * \text{der\_parz}[j]$

  $t \leftarrow t + 1$

AdaBOOST, algoritmo

for $i = 1, 2, \dots, N$

$\alpha_i = \frac{1}{N}$

for $t = 1, 2, \dots, T$

apprendi $f_t(x)$ con i pesi α_i (minimizza la f di costo)

calcola il coeff $\hat{\omega}_t$

ricalcola i pesi α_i

normalizza i pesi α_i

calcola la predizione finale $\hat{y} = \text{sign}\left[\sum_{t=1}^{T} \hat{\omega}_t f_t(x)\right]$

calcolo di $\hat{\omega}_t$: $\hat{\omega}_t = \frac{1}{2} \ln\left(\frac{1 - \text{weighted\_error}(f_t)}{\text{weighted\_error}(f_t)}\right)$

$$\text{weighted\_error}(f_t) = \frac{\text{peso totale errori}}{\text{peso totale di tutti i data points}} = \frac{\sum_{i=1}^{N} \alpha_i I[\hat{y}_i \neq y_i]}{\sum_{i=1}^{N} \alpha_i}$$

calcolo dei pesi: $\alpha_i \leftarrow \begin{cases} \alpha_i \cdot e^{-\hat{\omega}_t} & \text{se } f_t(x_i) = y_i \\ \alpha_i \cdot e^{\hat{\omega}_t} & \text{se } f_t(x_i) \neq y_i \end{cases}$

normalizzazione dei pesi: $\alpha_i \leftarrow \frac{\alpha_i}{\sum_{j=1}^{N} \alpha_j}$

classification

Quality Maximization:

Qualità totale con L2 penalty: $\text{Qualità\_Totale\_L2} = \ln \mathcal{L}(w) - \lambda \|w\|_2^2$

↳ se $\lambda = 0$: max likelihood $\rightarrow \hat{w}_{MLE}$

↳ se $\lambda \to \infty$: $\forall$ soluz con $\hat{w} \neq 0$, qualità $\to -\infty$. unica soluz $\hat{w} = 0$

↳ se $0 < \lambda < \infty$: $0 < \|\hat{w}\|_2^2 < \|\hat{w}_{MLE}\|_2^2$

Gradient Ascent

$$\frac{\partial \text{Qualità\_Totale\_L2}(w^{(t)})}{\partial w_j} = \text{deriv\_part}[j] - 2\lambda w_j^{(t)}$$

algoritmo

$w^{(1)} = 0$

$t = 1$

while $\|\nabla \text{Qualità\_Totale\_L2}(w^{(t)})\|_2 > \varepsilon$

for $j = 0, 1, \dots, D$

$\text{deriv\_part}[j] = \sum_{i=1}^{N} \Phi_j(x_i) \{ I[y_i = +1] - P(y_i = +1 \mid x_i, w^{(t)}) \}$

$w_j^{(t+1)} \leftarrow w_j^{(t)} - \alpha * \text{deriv\_part}[j] - 2\lambda w_j^{(t)}$

$t \leftarrow t + 1$

Clustering

K-means clustering, algoritmo:

scelta di k

inizializzazione di $\mu_1, \dots, \mu_k$

while not converged:

for $i = 1, \dots, N$

$z_i \leftarrow \underset{j}{\text{argmin}} \|\mu_j - x_i\|^2$ // z_i = etichetta del cluster a cui è x_i

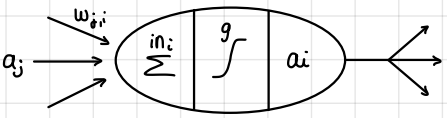
for $j = 1, \dots, k$

$\mu_j = \frac{1}{n_j} \sum_{i : z_i = j} x_i$ // n_j = # elementi cluster

K-means ++ clustering, algoritmo di smart initialization:

1. scegliere il 1° centroide in modo casuale tra tutti i data points
2. $\forall x_i$ calcolare la distanza $d(x_i)$ dal più vicino centroide
3. scegliere il NUOVO centroide tra i data point, con la probabilità di scegliere x_i proporzionale alla distanza tra x_i ed il centroide più vicino GIÀ SCELTO
4. Ripetere gli STEP 2, 3 fino ad avere k centroidi

Reti Neurali



$a_i = g(in_i)$

$in_i = \sum_{j=1}^{n} w_{j,i} a_j$

possibili g:

sigmoide $g(in_i) = \frac{1}{1 + e^{-in_i}}$

gradino $g(in_i) = \begin{cases} 1 & \text{se } in_i > t \\ 0 & \text{else} \end{cases}$

tangente iperbolica $g(in_i) = \tanh(in_i)$

ReLU $g(in_i) = \max(in_i, 0)$

se si fa uso della funzione gradino si sostituisce alla soglia t un ingresso aggiuntivo con valore $a_0 = -1$ e peso $w_{0,i} = t$

un perceptrone (rete feed forward a strato singolo) con funzione di attivazione a maggioranza ha tutti i pesi $W_j = 1$ e $W_0 = \frac{n}{2}$. la funzione è definita come: $g\left(\sum_{i=1}^{n} x_i - \frac{n}{2}\right)$ cioè $g > 0$ sse $\sum_{i=1}^{n} x_i > \frac{n}{2}$. Un perceptrone a soglia così definito fa parte dei separatori lineari

ALGORITMO di Apprendimento Perceptrone a sigmoide

Funzione di COSTO: $E = \sum_{\text{training}} \frac{1}{2} Err^2 = \sum_{\text{training}} \frac{1}{2} (y - f_w(x))^2$

Aggiornamento: $w_j \leftarrow w_j + \Delta w_j = w_j - \alpha \frac{\partial E}{\partial w_j}$

Derivata Parziale: $\frac{\partial E}{\partial w_j} = -Err \times g'(in) \times x_j$ // per la sigmoide $g' = g(1-g)$

// per perceptroni a soglia g' è indefinita e si omette

Gradient Descent, algoritmo

function APPRENDIMENTO_PERCETTRONE (esempi, rete)

returns ipotesi percettrone

repeat

for each e in esempi do:

$in \leftarrow \sum_{j=0}^{n} w_j x_j[e]$

$Err \leftarrow y[e] - g(in)$

$w_j \leftarrow w_j + \alpha \times Err \times g'(in) \times x_j[e]$

until termination_criteria

return IPOTESI_RETE_NEURALE (rete)

Algoritmo di Apprendimento Percettrone Multistrato

Funzione di costo: $E = \frac{1}{2} \sum_i (y_i - a_i)^2 = \frac{1}{2} \sum_i Err_i^2$ per un singolo punto

Aggiornamento: $w_{j,i} \leftarrow w_{j,i} + \Delta w_{j,i} = w_{j,i} - \alpha \frac{\partial E}{\partial w_{j,i}}$

Derivata Parziale: $\frac{\partial E}{\partial w_{j,i}} = -a_j \Delta_i$, con $\Delta_i = Err_i \times g'(in_i)$ errore modificato

Per lo strato nascosto: $w_{k,j} \leftarrow w_{k,j} + \Delta w_{k,j}$

$\Delta w_{kj} = -\alpha \frac{\partial E}{\partial w_{k,j}} = -\alpha (-a_k \Delta_j)$

$\Delta_j = g'(in_i) \sum_i w_{j,i} \Delta_i$

$w_{k,j} \leftarrow w_{k,j} + \alpha \times a_k \times \Delta_j$