Machine Learning Foundations

No.

Date . .

Perceptron: $H = \text{sign}(W^T X)$

train: if $\text{sign}(W_t^T X_n) \neq y_n$

$W_{t+1} \leftarrow W_t + y_n X_n$

$$W_f^T W_{t+1} = W_f^T (W_t + y_n(t) X_n(t))$$
$$\geq W_f^T W_t + \min_n y_n W_f^T X_n$$
$$> W_f^T W_t + 0$$

$$\|W_{t+1}\|^2 = \|W_t + y_n(t) X_n(t)\|^2$$
$$= \|W_t\|^2 + 2 y_n(t) W_t X_n(t) + \|y_n(t) X_n(t)\|^2$$

$\because y_n(t) W_t^T X_n(t) < 0$ $\therefore \leq \|W_t\|^2 + 0 + \|y_n(t) X_n(t)\|^2$

$\because y_n(t) = \pm 1$ $\leq \|W_t\|^2 + \max_n \|X_n\|^2$

$$\frac{W_f^T W_T}{\|W_f\| \|W_T\|} \geq \sqrt{T} \cdot \text{constant}$$

$$\|W_1\|^2 \leq \|W_0\|^2 + \max_n \|X_n\|^2 \leq \cdots \|W_{T-1}\|^2 + \max_n \|X_n\|^2 \cdot T$$
$$\leq T \cdot \max_n \|X_n\|^2$$

$$\|W_T\| \leq \sqrt{T} \max_n \|X_n\|$$

$$W_f^T W_T \geq W_f^T W_{T-1} + \min_n y_n W_f^T X_n \geq \cdots W_f^T W_0 + T \cdot \min_n y_n W_f^T X_n$$

$$W_f^T W_T \leq T \cdot \min_n y_n W_f^T X_n$$

$$\therefore \frac{W_f^T W_T}{\|W_f\| \|W_T\|} \geq \frac{T \cdot \min_n y_n W_f^T X_n}{\|W_f\| \sqrt{T} \max_n \|X_n\|^2}$$

$$\text{constant} = \frac{\min_n y_n W_f^T X_n}{\|W_f\| \max_n \|X_n\|^2}$$

$$\frac{W_f^T W_T}{\|W_f\| \|W_T\|} \leq 1$$

No.

Date

Hoeffding's Inequality: in big sample (N large), $\nu \approx \mu$ (within ε)

$$P[|\nu-\mu|>\varepsilon] \le 2e^{(-2\varepsilon^2 N)}$$

$$P(BAD) = P(|E_{in}-E_{out}|>\varepsilon) \le 2m_H(2N)P[|E_{in}(h)-E_{in}'(h)|>\frac{\varepsilon}{2}]$$

$$\le 2m_H(2N)\cdot 2e^{-2(\frac{\varepsilon}{4})^2 N} = 4m_H(2N)e^{-\frac{1}{8}\varepsilon^2 N}$$

$$\because |E_{in}-E_{in}'|>\frac{\varepsilon}{2} \iff |E_{in}-\frac{E_{in}+E_{in}'}{2}|>\frac{\varepsilon}{4}$$

$$P(|E_{in1}-E_{out1}|>\varepsilon \text{ or } |E_{in2}-E_{out2}|>\varepsilon \text{ or } \cdots |E_{inM}-E_{outM}|>\varepsilon)$$

VC bound

$$m_H(N) \le B(N,k) < \sum_{i=0}^{k-1} C_N^i$$

$B(N,k)$: Break Point — the one can't shatter k points ($<2^k$)

highest term N^{k}

N^{dvc}

VC dimension of $\mathcal{H}$, denoted $d_{vc}(\mathcal{H})$ is

largest N for which $m_H(N) = 2^N$

$d_{vc} = k_{min} - 1$

finite $d_{vc} \Rightarrow g$ will generalize ($E_{out}(g) \approx E_{in}(g)$)

d-D perceptron $d_{vc} = d+1$

$d_{vc} \approx$ #free parameters

No.

Date . .

$$E_{out}(g) \le E_{in}(g) + \underbrace{\sqrt{\frac{8}{N}\ln\left(\frac{4(2N)^{d_{vc}}}{\delta}\right)}}_{\Omega(N, \mathcal{H}, \delta)}$$

$d_{vc}\uparrow$, $E_{in}\downarrow$, $\Omega\uparrow$

$d_{vc}\downarrow$, $\Omega\downarrow$, $E_{in}\uparrow$

$$P_D[|E_{in}(g) - E_{out}(g)| > \varepsilon] \le \delta$$

BAD

Linear Regression

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

$$E_{in}(\vec{w}) = \frac{2}{N}(X^TX\vec{w} - X^T\vec{y}) = 0 \rightarrow \vec{w} = \underline{X^\dagger}\vec{y}$$

$\downarrow$

pseudo-inverse

Logistic Regression

$$h(x) = \theta(w^Tx) = \frac{1}{1+e^{(-w^Tx)}}$$

likelihood → the possibility of generating the data by a specific hypothesis

$$= P(x_1)h(x_1) * P(x_2)(1-h(x_2)) \cdots * P(x_n)(1-h(x_n)) = \prod_{n=1}^{N}\theta(y_n\vec{w}^Tx_n)$$

Cross-Entropy Error : $\max_{\vec{w}} \text{likelihood}(\vec{w}) \propto \prod_{n=1}^{N}\theta(y_n\vec{w}^Tx_n)$

$\Downarrow$

$$\max_{\vec{w}} \sum \ln\theta(y_n\vec{w}^Tx_n)$$

$\Downarrow$

$$\min_{\vec{w}} \frac{1}{N}\sum_{n=1}^{N} -\ln\theta(y_n\vec{w}^Tx_n)$$

No.

Date

$$E_{in}(\vec{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln\left(1+e^{(-y_n \vec{w}^T x_n)}\right)$$

$$\nabla E_{in}(\vec{w}) = \frac{1}{N}\sum_{n=1}^{N} \theta(-y_n \vec{w}^T x_n)(-y_n x_n)$$

Gradient Descent

$$\vec{w}_{t+1} \leftarrow \vec{w}_t - \eta \frac{\nabla E_{in}(\vec{w}_t)}{\|\nabla E_{in}(\vec{w}_t)\|}$$

Linear Models for Classification

Stochastic Gradient Descent

$$\vec{w}_{t+1} \leftarrow \vec{w}_t + \eta \underbrace{\theta(-y_n \vec{w}_t^T x_n)(y_n x_n)}_{-\nabla err(\vec{w}_t, x_n, y_n)}$$

Multiclass { One-vs-ALL / One-vs-One

Nonlinear Transformation

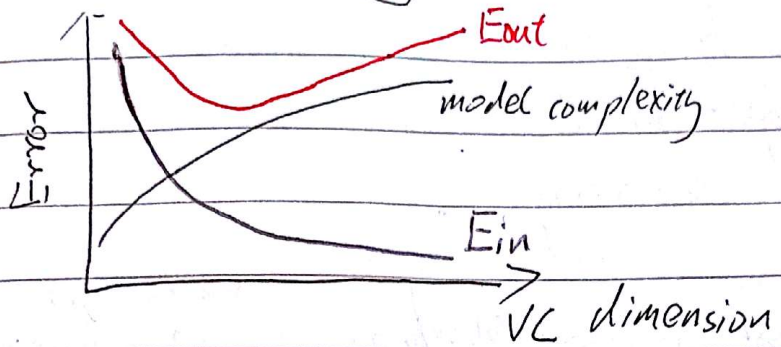
$$\Phi_Q(x) = (1,\ x_1, x_2 \cdots x_d,\ x_1^2, x_1x_2, \cdots x_d^2,\ \cdots\ x_1^Q, x_1^{Q-1}x_2, \cdots, x_d^Q)$$

$O(Q^d)$

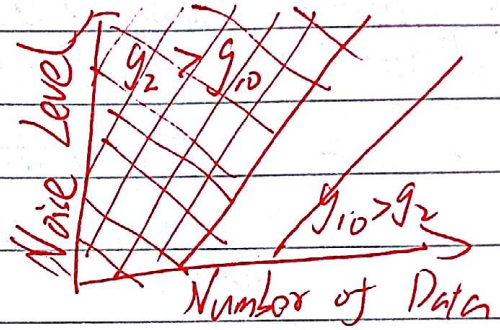
No.

Date . .

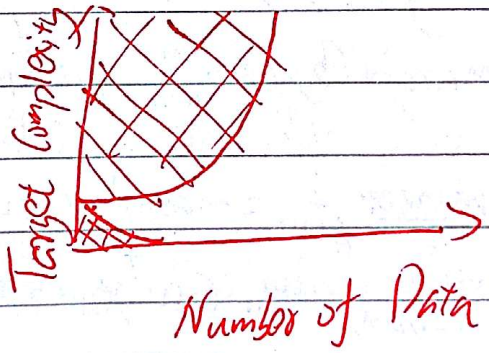
Hazard of Overfitting



stochastic noise



deterministic noise



Regularization

$\lambda > 0$ $\lambda \uparrow$ $w \downarrow$ $c \downarrow$

→ Lagrange Multiplier

Ridge Regression

$\nabla E_{in}(\vec{w}_{REG}) + \frac{2\lambda}{N}\vec{w}_{REG} = 0$

optimal solution:

$\vec{w}_{REG} \leftarrow (Z^T Z \vec{w}_{REG} - Z^T y) + \lambda \vec{w}_{REG} = 0$

$\vec{w}_{REG} \leftarrow (Z^T Z + \lambda I)^{-1} Z^T y$

→ $\min E_{in}(\vec{w}) + \frac{\lambda}{N}\vec{w}^T\vec{w}$

No.

Date

Three Learning Principles

① Occam's Razor
② Sampling Bias
③ Data snooping

Linear Support Vector Machine

core idea: make the margin as large as possible

Locate specific specially-scaled (b, w)

$$\text{distance}(x, h) = \text{distance}(x, b, \vec{w})$$

$$= \frac{1}{\|w\|}(\vec{x}\vec{w} + b) \quad \frac{1}{\|w\|}(\vec{w}^T x + b)$$

$$= \frac{1}{\|w\|} y_n (w^T x_n + b)$$

$$\max_{b,w} \text{margin}(b, w)$$

$$\text{margin}(b, w) = \min_{n=1,2,\dots,N} \frac{1}{\|w\|} y_n (w^T x_n + b)$$

$\Downarrow$ special scaling

$$\max_{b,w} \frac{1}{\|w\|} = \text{margin}(b, w)$$

$$\min_{n=1,2,\dots,N} y_n (w^T x_n + b) = 1$$

$\Rightarrow$

$$\min_{b,w} \frac{1}{2} w^T w$$

$$y_n (w^T x_n + b) \geq 1$$

for all n

this is because if the max function to $\frac{1}{\|w\|}$ will push the $y_n(w^T x_n + b)$'s min to be 1.

$$u = \begin{bmatrix} b \\ w \end{bmatrix}; \quad Q = \begin{bmatrix} 0 & 0_d^T \\ 0_d & I_d \end{bmatrix}; \quad p = 0_{d+1}$$

$$a_n^T = y_n [1 \;\; x_n^T]; \quad c_n = 1; \quad M = N$$

Quadratic Programming,

Optimal $u \leftarrow QP(Q, p, A, c)$

$$\min_u \frac{1}{2} u^T Q u + p^T u$$

subject to $a_m^T u \geq c_m$

for $m = 1, 2, \dots, M$

No.

Date . .

Dual Support Vector Machine

view λ's in regularization as unknown given the constraints, and solve them as variables instead.

locate support vector (z_n, y_n) & their α_n

Lagrange Function with Lagrange Multipliers α_n (replace λ_n)

$$\mathcal{L}(b, w, \alpha) = \frac{1}{2} w^T w + \sum_{n=1}^{N} \alpha_n (1 - y_n (w^T z_n + b))$$

($\frac{1}{2} w^T w$: objective; $1 - y_n(w^T z_n + b)$: constraint)

$$SVM \equiv \min_{b,w} \left(\max_{\text{all } \alpha_n \geq 0} \mathcal{L}(b, w, \alpha) \right)$$

solve this

@ any 'violating' (b, w): $\max_{\text{all } \alpha_n \geq 0} \left(\frac{1}{2} w^T w + \sum_n \alpha_n (\text{positive}) \right) \to \infty$

@ any 'feasible' (b, w): $\max_{\text{all } \alpha_n \geq 0} \left(\frac{1}{2} w^T w + \sum_n \alpha_n (\text{all non-positive}) \right) = \frac{1}{2} w^T w$

$\therefore \min_{b,w} \left(\max_{\text{all } \alpha_n \geq 0} \mathcal{L}(b, w, \alpha) \right) = \min_{b,w} \left(\infty \text{ if violate; } \frac{1}{2} w^T w \text{ if feasible} \right)$

Lagrange Dual Problem:

$$\min_{b,w} \left(\max_{\text{all } \alpha_n \geq 0} \mathcal{L}(b, w, \alpha) \right) \geq \max_{\text{all } \alpha_n \geq 0} \left(\min_{b,w} \mathcal{L}(b, w, \alpha') \right)$$

$\geq$: weak duality

"$=$": strong duality, true for QP if

① convex primal ② feasible primal ③ linear constraints

Simplifications

$$(\text{gradience of } b = 0) \Rightarrow \max_{\text{all } \alpha_n \geq 0, \sum y_n \alpha_n = 0} \left(\min_{w} \frac{1}{2} w^T w + \sum_{n=1}^{N} \alpha_n (1 - y_n (w^T z_n)) \right)$$

$$(\text{gradience of } w = 0) \Rightarrow \max_{\text{all } \alpha_n \geq 0, \sum y_n \alpha_n = 0, w = \sum \alpha_n y_n z_n} \left(-\frac{1}{2} \left\| \sum_{n=1}^{N} \alpha_n y_n z_n \right\|^2 + \sum_{n=1}^{N} \alpha_n \right)$$

$\Downarrow$

Dual SVM:

$$\min_{\alpha} \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m z_n^T z_m - \sum_{n=1}^{N} \alpha_n$$

subject to $\sum_{n=1}^{N} y_n \alpha_n = 0$

$\alpha_n \geq 0$, for $n = 1 \ldots N$.

No.

Date

because, when violate the condition $y_n(w^T z_n + b) \geq 1$, α_n will be $+\infty$ which will be filtered by min function. when fit the condition, α_n will be 0 or $y_n(w^T z_n + b) = 1$

KKT conditions

if primal-dual optimal (b, w, α)

① primal feasible: $y_n(w^T z_n + b) \geq 1$

② dual feasible: $\alpha_n \geq 0$

③ dual-inner optimal: $\sum y_n \alpha_n = 0$; $w = \sum \alpha_n y_n z_n$

④ primal-inner optimal: $\alpha_n(1 - y_n(w^T z_n + b)) = 0$

Kernel Support Vector Machine

embed ϕ into kernel

using kernel to replace ~~$\phi(x, x')$~~ $\phi(x)^T \phi(x')$

so that we can avoid computation of z.

e.g. : $\phi_2(x) = (1, x_1, \ldots, x_d, x_1^2, \ldots x_d^2)$

$$K_{\phi_2}(x, x') = 1 + x^T x' + (x^T x')^2$$

$$K_2(x, x') = 1 + 2\gamma x^T x' + \gamma^2 (x^T x')^2 = (1 + \gamma x^T x')^2 \quad \text{where } \gamma > 0$$

~~Kernel~~ Kernel $\longleftrightarrow$ definition of margin

General Polynomial Kernel $K_Q(x, x') = (\zeta + \gamma x^T x')^Q$

where $\gamma > 0$, $\zeta \geq 0$

Gaussian Kernel $K(x, x') = \exp(-\gamma \|x - x'\|^2)$

Radial Basis Function (RBF)

$$g_{SVM}(x) = \text{sign}\left(\sum_{SV} \alpha_n y_n \exp(-\gamma \|x - x_n\|^2) + b\right)$$

large γ $\Rightarrow$ sharper Gaussians $\Rightarrow$ overfitting

No.

Date . .

Kernel represents special similarity

Soft-Margin Support Vector Machine

in order to suit soft-margin, choose to loose the conditions of the optimization target.

$$\begin{cases} \min\limits_{b,w} \ \frac{1}{2}w^Tw + C\cdot\sum\limits_{n=1}^{N}[\![y_n \neq \text{sign}(w^Tz_n+b)]\!] \\ \text{s.t.} \quad y_n(w^Tz_n+b) \geq 1 - \infty\cdot[\![y_n \neq \text{sign}(w^Tz_n+b)]\!] \end{cases}$$

C: the trade-off of large margin & noise tolerance

$\infty\cdot[\![\cdot]\!]$: when the data is not correct

non-linear → not QP

record margin violation by ξ_n

penalize with margin violation

$$\min_{b,w,\xi} \ \frac{1}{2}w^Tw + C\cdot\sum_{n=1}^{N}\xi_n$$

$$\text{s.t.} \quad y_n(w^Tz_n+b) \geq 1-\xi_n \text{ and } \xi_n \geq 0 \text{ for all } n$$

$\begin{cases} C\uparrow, \text{ margin violation}\downarrow \\ C\downarrow, \text{ margin}\uparrow \end{cases}$

⇓ Lagrange Dual

$$\mathcal{L}(b,\vec{w},\vec{\xi},\vec{\alpha},\vec{\beta}) = \underbrace{\frac{1}{2}w^Tw + C\cdot\sum_{n=1}^{N}\xi_n}_{\text{objective}} + \underbrace{\sum_{n=1}^{N}\alpha_n\cdot(1-\xi_n-y_n(w^Tz_n+b)) + \sum_{n=1}^{N}\beta_n(-\xi_n)}_{\text{constrains}}$$

$$\max_{\alpha_n\geq 0,\beta_n\geq 0}\left(\min_{b,w,\xi}\mathcal{L}(b,w,\xi,\alpha,\beta)\right)$$

⇓ $\frac{\partial\mathcal{L}}{\partial\xi_n} = 0 = C-\alpha_n-\beta_n \Rightarrow \begin{matrix}\beta_n = \text{自}C-\alpha_n \\ 0\leq\alpha_n\leq C\end{matrix}$

$$\max_{0\leq\alpha_n\leq C,\beta_n=C-\alpha_n}\left(\min_{b,w}\frac{1}{2}w^Tw + \sum_{n=1}^{N}\alpha_n(1-y_n(w^Tz_n+b))\right)$$

then use $\frac{\partial\mathcal{L}}{\partial b}=0$ and $\frac{\partial\mathcal{L}}{\partial w}=0$ to simplify

No.

Date . .

$$\min_{\alpha} \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m y_n y_m z_n^T z_m - \sum_{n=1}^{N}\alpha_n$$

$$\text{s.t.} \sum_{n=1}^{N} y_n\alpha_n = 0; \quad 0 \le \alpha_n \le C \text{ for all } n$$

implicity $w = \sum_{n=1}^{N}\alpha_n y_n z_n$; $\beta_n = C - \alpha_n$ for all n.

free SV → $(0 < \alpha_n < C)$: $\xi_n = 0$ → on fat boundary → locates b

non SV → $(0 = \alpha_n)$: $\xi_n = 0$ → away from fat boundary

bounded SV → $(\alpha_n = C)$: ξ_n = violation amount (→ violate fat boundary)

complementary slackness:

$$\alpha_n(1 - \xi_n - y_n(w^T z_n + b)) = 0$$

$$(C - \alpha_n)\xi_n = 0$$

Kernel Logistic Regression

two-level learning combining SVM and logistic regression

$$g(x) = \theta(A \cdot (w_{SVM}^T \phi(x) + b_{SVM}) + B)$$

often $A > 0$ if w_{SVM} good

$B \approx 0$ if w_{SVM} good.

$$\min_{A,B} \frac{1}{N}\sum_{n=1}^{N} \log\left(1 + \exp\left(-y_n(A \cdot (w_{SVM}^T \phi(x_n) + b_{SVM}) + B)\right)\right)$$

Probabilistic SVM

Kernel SVM ⇒ approx. LogReg in Z-space

solving L2-regularized logistic regression

$$\min_{w} \frac{\lambda}{N} w^T w + \frac{1}{N}\sum_{n=1}^{N} \log(1 + \exp(-y_n w^T z_n))$$

yields optimal solution $w_* = \sum_{n=1}^{N} \beta_n z_n$ — theorem

representer theory

No.

Date . .

without loss of generality, can solve for optimal β instead of w

$$\min_{\beta} \frac{\lambda}{N} \sum_{n=1}^{N} \sum_{m=1}^{N} \beta_n \beta_m K(x_n, x_m) + \frac{1}{N} \sum_{n=1}^{N} \log\left(1 + \exp\left(-y_n \sum_{m=1}^{N} \beta_m K(x_n, x_m)\right)\right)$$

Support Vector Regression

Kernel Ridge Regression Problem — non-linear regression

$$\min_{w} \frac{\lambda}{N} w^T w + \frac{1}{N} \sum_{n=1}^{N} (y_n - w^T z_n)^2$$

yields optimal solution $w_* = \sum_{n=1}^{N} \beta_n z_n$

without loss of generality, can solve for optimal β instead of w

$$\min_{\beta} \frac{\lambda}{N} \sum_{n=1}^{N} \sum_{m=1}^{N} \beta_n \beta_m K(x_n, x_m) + \frac{1}{N} \sum_{n=1}^{N} \left(y_n - \sum_{m=1}^{N} \beta_m K(x_n, x_m)\right)^2$$

$$\nabla E_{aug}(\beta) = \frac{2}{N}\left(\lambda K^T I \beta + K^T K \beta - K^T y\right) = \frac{2}{N} K^T \left((\lambda I + K)\beta - y\right)$$

$$\nabla E_{aug}(\beta) = 0 \Rightarrow \beta = (\lambda I + K)^{-1} y$$

for classification

Least-Squares SVM — dense β

Tube Regression (L2 regularized) { no error if within a tube; error by distance to tube if outside a tube

error measure: $err(y, s) = \max(0, |s-y| - \varepsilon)$ { $|s-y| \le \varepsilon : 0$; $|s-y| \ge \varepsilon : |s-y| - \varepsilon$

ε-insensitive error $\quad \varepsilon > 0$

↓ mimic standard SVM

Standard Support Vector Regression Primal

$$\min_{b, w, \xi_n^\vee, \xi_n^\wedge} \frac{1}{2} w^T w + C \sum_{n=1}^{N} (\xi_n^\vee + \xi_n^\wedge)$$

$$\text{s.t. } -\varepsilon - \xi_n^\vee \le y_n - w^T z_n - b \le \varepsilon + \xi_n^\wedge$$

$$\xi_n^\wedge, \xi_n^\vee \ge 0$$

No.

Date . .

C: trade-off of regularization & tube violation

ε: vertical tube width

introduce Lagrange Multipliers $\alpha^\wedge$ & $\alpha^\vee$

$$\min \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}(\alpha_n^\wedge-\alpha_n^\vee)(\alpha_m^\wedge-\alpha_m^\vee)K_{n,m}+\sum_{n=1}^{N}((\varepsilon-y_n)\cdot\alpha_n^\wedge+(\varepsilon+y_n)\cdot\alpha_n^\vee)$$

$$\text{s.t. } \sum_{n=1}^{N}1\cdot(\alpha_n^\wedge-\alpha_n^\vee)=0$$

$$0\le\alpha_n^\wedge\le C,\ 0\le\alpha_n^\vee\le C$$

$$w=\sum_{n=1}^{N}(\alpha_n^\wedge-\alpha_n^\vee)z_n$$

complementary slackness: $\alpha_n^\wedge(\varepsilon+\xi_n^\wedge-y_n+w^Tz_n+b)=0$

$\alpha_n^\vee(\varepsilon+\xi_n^\vee+y_n-w^Tz_n-b)=0$

within tube $|w^Tz_n+b-y_n|<\varepsilon \Rightarrow \alpha_n^\wedge,\alpha_n^\vee=0$

Blending and Bagging

linear blending: known g_t, each to be given α_t ballot

$$G(x)=\text{sign}\left(\sum_{t=1}^{T}\alpha_t\cdot g_t(x)\right) \text{ with } \alpha_t\ge 0$$

any blending (stacking)

BAGging (Bootstrap Aggregation)

Adaptive Boosting

$u^{(1)}=\frac{1}{N}$

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

① obtain g_t by A(D, $u^{(t)}$), where A tries to minimize $u^{(t)}$ weighted 0/1 error.

② update $u^{(t)}$ to $u^{(t+1)}$ by $\Delta_t=\sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}}$, where ε_t = weighted error rate of g_t.

~~$G(x)$ = combine all g_t linear or non-linear~~ $u_n^{(t+1)}=u_n^{(t)}\cdot\Delta_t^{-y_ng_t(x_n)}$

No.

Date . .

voting weight

② compute $\alpha_t = \ln(\Delta_t)$

return $G(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t g_t(x)\right)$

Decision Tree

4 choices: ① # of branches; ② branching criteria; ③ termination criteria; ④ base hypothesis

CART: human-explainable; multiclass; categorical features easily; missing features easily; efficient non-linear training

↳ using Gini Impurity: as the ~~diff~~ criteria

↳ $1 - \sum_{k=1}^{K} \left(\frac{\sum_{n=1}^{N} [\![y_n = k]\!]}{N}\right)^2$

Regression Error: $\text{impurity}(D) = \frac{1}{N}\sum_{n=1}^{N} (y_n - \bar{y})^2$

Pruning { $\arg\min_{\text{all possible } G} E_{in}(G) + \lambda \Omega(G)$ → # of leaves

try to remove each leaf and calculate the ~~formula~~ formula above to select the best

Random Forest

of trees ↑, smoothness ↑ & large-margin ↑

bagging + random combination CART

noise can be corrected by "voting"!

Feature Importance by Permutation Test

use random shuffle to one feature, ~~to~~ to degrade this feature to test whether the ~~class~~ performance get worse or not

Gradient Boosted Decision Tree

AdaBoost + decision tree: always use sampling $\propto u_n^{(t)}$ + pruned DTree($\tilde{D}$)

In AdaBoost $u_n^{(t+1)} = \begin{cases} u_n^{(t)} \cdot \Delta_t & \text{if incorrect} \\ u_n^{(t)} / \Delta_t & \text{if correct} \end{cases} = u_n^{(t)} \cdot e^{(-y_n \alpha_t g_t(x_n))}$

No.

Date

$$u_n^{(T+1)} = u_n^{(1)} \cdot \prod_{t=1}^{T} e^{(-y_n \alpha_t g_t(x_n))} = \frac{1}{N} \cdot e^{(-y_n \sum_{t=1}^{T} \alpha_t g_t(x_n))}$$

∴ AdaBoost: $u_n^{(T+1)} \propto \exp(-y_n \cdot (\text{voting score on } x_n))$

AdaBoost decreases $\sum_{n=1}^{N} u_n^{(t)}$ and thus somewhat minimizes

$$\sum_{n=1}^{N} u_n^{(T+1)} = \frac{1}{N} \sum_{n=1}^{N} \exp(-y_n \sum_{t=1}^{T} \alpha_t g_t(x_n))$$

< recall that $G(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t g_t(x)\right)$

linear score $s = \sum_{t=1}^{T} \alpha_t g_t(x_n)$

$\text{err}_{0/1}(s, y) = [\![ys \le 0]\!]$

$\widehat{\text{err}}_{\text{ADA}}(s, y) = \exp(-ys)$ ⟩ upper bound of $\text{err}_{0/1}$

exponential error measure

Use Gradient Descent to ~~solve~~ minimize $\widehat{\text{err}}_{\text{ADA}}$

at iteration t, to find g_t, solve (looks $h(x_n)$ as a step towards the valley)

$$\min_h \widehat{E}_{\text{ADA}} = \frac{1}{N} \sum_{n=1}^{N} \exp(-y_n (\sum_{\tau=1}^{t-1} \alpha_\tau g_\tau(x_n) + \eta h(x_n)))$$

$$= \sum_{n=1}^{N} u_n^{(t)} \exp(-y_n \eta h(x_n))$$

taylor ↘

$$\approx \sum_{n=1}^{N} u_n^{(t)} (1 - y_n \eta h(x_n)) = \sum_{n=1}^{N} u_n^{(t)} - \eta \sum_{n=1}^{N} u_n^{(t)} y_n h(x_n)$$

so we want a good h (function direction) $\leftrightarrow \min \sum_{n=1}^{N} u_n^{(t)} (-y_n h(x_n))$

$$\sum_{n=1}^{N} u_n^{(t)} (-y_n h(x_n)) = \sum_{n=1}^{N} u_n^{(t)} \begin{cases} -1 & \text{if } y_n = h(x_n) \\ +1 & \text{if } y_n \neq h(x_n) \end{cases}$$

$$= -\sum_{n=1}^{N} u_n^{(t)} + \sum_{n=1}^{N} u_n^{(t)} \begin{cases} 0 & \text{if } y_n = h(x_n) \\ 2 & \text{if } y_n \neq h(x_n) \end{cases}$$

$$= -\sum_{n=1}^{N} u_n^{(t)} + 2 E_{in}^{u^{(t)}}(h) \cdot N$$

A in AdaBoost minimize $E_{in}^{u^{(t)}}(g_t)$

AdaBoost finds g_t by approximately $\min_h \widehat{E}_{ADA} = \sum_{n=1}^{N} u_n^{(t)} \exp(-y_n \eta h(x_n))$

after finding g_t, ~~how about~~ find best η to $\min \widehat{E}_{ADA}$

steepest descent

No.

Date

weighted error rate

$$\begin{cases} y_n = g_t(x_n) : u_n^{(t)} \exp(-\eta) & \text{correct} \\ y_n \neq g_t(x_n) : u_n^{(t)} \exp(+\eta) & \text{incorrect} \end{cases}$$

then $\hat{E}_{ADA} = \left(\sum_{n=1}^{N} u_n^{(t)}\right) \cdot \left((1-\varepsilon_t)\exp(-\eta) + \varepsilon_t \exp(+\eta)\right)$

by solving $\frac{\partial \hat{E}_{ADA}}{\partial \eta} = 0$, steepest $\eta_t = \ln\sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}} = \alpha_t$ → voting weighted

∴ AdaBoost: steepest descent with approximate functional gradient

$$\min_{\eta} \min_{h} \frac{1}{N} \sum_{n=1}^{N} \exp\left(-y_n\left(\sum_{\tau=1}^{t-1} \alpha_\tau g_\tau(x_n) + \eta h(x_n)\right)\right)$$

Gradient Boost: use the same thought, change the error function.

functional gradient descent

$$\min_{\eta} \min_{h} \frac{1}{N} \sum_{n=1}^{N} \mathrm{err}\left(\underbrace{\sum_{\tau=1}^{t-1} \alpha_\tau g_\tau(x_n)}_{s_n} + \eta h(x_n), y_n\right) \quad \text{with}$$

e.g. $\mathrm{err}(s, y) = (s-y)^2$ for regression problem

find $g_t = h$:

$$\overset{\text{taylor}}{\approx} \min_{h} \frac{1}{N} \sum_{n=1}^{N} \underbrace{\mathrm{err}(s_n, y_n)}_{\text{constant}} + \frac{1}{N} \sum_{n=1}^{N} \eta h(x_n) \left.\frac{\partial \mathrm{err}(s, y_n)}{\partial s}\right|_{s=s_n}$$

$$= \min_{h} \text{constant} + \frac{\eta}{N} \sum_{n=1}^{N} h(x_n) \cdot 2(s_n - y_n)$$

add $(h(x_n))^2$ to penalize the $\pm\infty$ choice, since the magnitude of h doesn't matter, because η will be optimized little later.

$$\therefore = \min_{h} \text{constants} + \frac{\eta}{N} \sum_{n=1}^{N} \left((s_n - y_n)^2 + 2h(x_n)(s_n - y_n) + (h(x_n))^2\right)$$

$$= \min_{h} \text{constants} + \frac{\eta}{N} \sum_{n=1}^{N} \left(\text{constant} + (h(x_n) - (y_n - s_n))^2\right)$$

solution of penalized approximate functional gradient: squared-error regression on $\{(x_n, \underbrace{y_n - s_n}_{\text{residual}})\}$

find η:

$$\min_{\eta} \frac{1}{N} \sum_{n=1}^{N} (s_n + \eta h(x_n) - y_n)^2 = \frac{1}{N} \sum_{n=1}^{N} (y_n - s_n) - \eta g_t(x_n))^2$$

No.

Date

Gradient Boosted Decision Tree (GBDT)

$s_1 = s_2 \cdots = s_N = 0$

for $t = 1, 2 \cdots T$

① obtain g_t by $A(\{(x_n, y_n - s_n)\})$ where A is a (squred-error) regression algorithm

② compute $\alpha_t = \text{OneVarLinearReg}(\{(g_t(x_n), y_n - s_n)\})$

③ update $s_n \leftarrow s_n + \alpha_t g_t(x_n)$

return $G(x) = \sum_{t=1}^{T} \alpha_t g_t(x)$

Neural Network

Machine Learning Techniques (last five chapters)

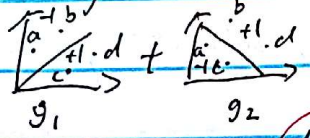
Neural Network

(1) every layer of can be seen as a linear combination of a sequence of perceptrons.

linear combination → AND, OR, NOT

(2) Choice of Activative Function: $\tanh(s) = \frac{\exp(s)-\exp(-s)}{\exp(s)+\exp(-s)} = 2\theta(s)-1$ (sigmoid)

(3) every layer can also be seen as a feature transform $\phi(x)$. To transform the original features list to the output of every hidden nodes.



$a(-1,-1), b(-1,+1), c(+1,+1), d(+1,-1)$

$$\phi^{(l)}(x) = \tanh\left(\begin{bmatrix} \sum_{i=0}^{d^{(l-1)}} w_{i1}^{(l)} x_i^{(l-1)} \\ \vdots \end{bmatrix}\right)$$

W and X more parallel, the value ↗

whether X 'matches' weight vectors in pattern

it is because $s_j^{(l)} = \sum_{i=0}^{d^{(l-1)}} w_{ij}^{(l)} x_i^{(l-1)}$

(4) $1 \le l \le L$, $0 \le i \le d^{(l-1)}$, $1 \le j \le d^{(l)}$

compute $\frac{\partial e_n}{\partial w_{ij}^{(l)}} = \frac{\partial e_n}{\partial s_j^{(l)}} \cdot \frac{\partial s_j^{(l)}}{\partial w_{ij}^{(l)}} = \delta_j^{(l)} \cdot (x_i^{(l-1)})$

$$\delta_j^{(l)} = \frac{\partial e_n}{\partial s_j^{(l)}} = \sum_{k=1}^{d^{(l+1)}} \frac{\partial e_n}{\partial s_k^{(l+1)}} \frac{\partial s_k^{(l+1)}}{\partial x_j^{(l)}} \frac{\partial x_j^{(l)}}{\partial s_j^{(l)}}$$

$$= \sum_{k=1}^{d^{(l+1)}} (\delta_k^{(l+1)})(w_{jk}^{(l+1)})(\tanh'(s_j^{(l)}))$$

the $\delta_j^{(l)}$ depends on the error function. e.g. squared err

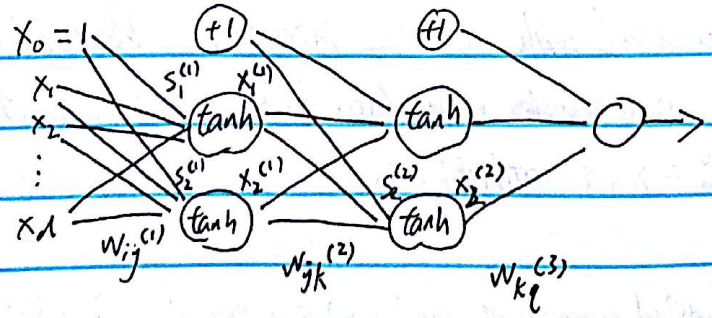
$\delta_j^{(l)} = -2(y_n - s_j^{(L)})$

(5) gradient descent: $w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} - \eta x_i^{(l-1)} \delta_j^{(l)}$

stochastic: every time use one data sample to do backprop

mini-batch: every time use several data samples, and use average $x_i^{(l-1)} \delta_j^{(l)}$ to update weights.

(6)



large w, shrink large
small w, shrink small

(7) $d_{vc} = O(VD)$ where V = # of neurons, D = # of weights

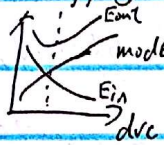
(8) Regularization: L_2 $\sum (w_{ij}^{(l)})^2$ → can't generate $w_{ij}^{(l)} = 0$ (sparse)

L_1 $\sum |w_{ij}^{(l)}|$ → not differentiable

weight-elimination ('scaled' L_2) regularizer: $\sum \frac{(w_{ij}^{(l)})^2}{1+(w_{ij}^{(l)})^2}$

make sure that large weights and small weights shrink at the same median scale.

(8) Early Stopping Regularizer (the more iterations, the larger d_{vc})



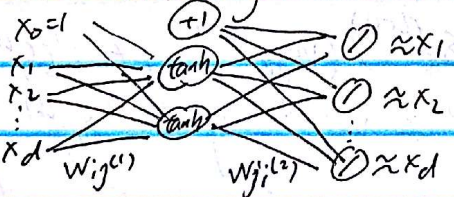
so, early stop optimization iteration

Deep Learning

(1) pre-train, to make the initial weights good for later train.

information-preserving encoding: next layer contains the same information with different representation.

(2) Information-Preserving Neural Network → Autoencoder



$d - \hat{d} - d$ NNet with goal $g_i(x) \approx x_i$

$w_{ij}^{(1)}$: encoding weights

$w_{ij}^{(2)}$: decoding weights

use squred error function $\sum_{i=1}^{d} (g_i(x) - x_i)^2$

sometimes constrain $w_{ij}^{(1)} = w_{ji}^{(2)}$ as regularization

(3) In order to deal with noise, introduce Denoising Autoencoder:

run basic autoencoder with data $\{(\tilde{x}_1, y_1 = x_1), (\tilde{x}_2, y_2 = x_2) \ldots (\tilde{x}_n, y_n = x_n)\}$

where $\tilde{x}_n = x_n$ + artificial noise noise-tolerant denoising a kind of regularization

(4) PCA (principle component analysis) ⇒ maximize $\sum$ (variance after projection)

linear autoencoder: maximum $\sum$(magnitude after projection)^2

linear hypothesis for k-th component $h_k(x) = \sum_{j=0}^{\tilde{d}} w_{jk}^{(2)} \left(\sum_{i=1}^{d} w_{ij}^{(1)} x_i \right)$

$\therefore w_{ij}^{(1)} = w_{ji}^{(1)} = w_{ij} = W$ (regularization)

$h_k(x) = W W^T x$

$\min_W E_{in}(h) = E_{in}(W) = \frac{1}{N} \sum_{n=1}^{N} \| x_n - W W^T x_n \|^2$ with $d \times \tilde{d}$ matrix W

eigen-decompose $WW^T = V \Gamma V^T$ where

$d \times d$ matrix V orthogonal : $VV^T = V^T V = I_d$

$d \times d$ matrix Γ diagonal with $\leq \tilde{d}$ non-zero

$\therefore \min_V \min_\Gamma \frac{1}{N} \sum_{n=1}^{N} \| V I V^T x_n - V \Gamma V^T x_n \|^2$

$V I V^T x_n \Rightarrow x_n$ because V and V^T represent rotate or reflect in the space, won't change length

$V \Gamma V^T x_n \Rightarrow W W^T x_n$

$\min_\Gamma \sum \| (I - \Gamma)(\text{some vector}) \|^2$

want as many 0 as possible

$\min_V \sum_{n=1}^{N} \left\| \begin{bmatrix} 0 & 0 \\ 0 & I_{d-\tilde{d}} \end{bmatrix} V^T x_n \right\|^2 \equiv \max_V \sum_{n=1}^{N} \left\| \begin{bmatrix} I_{\tilde{d}} & 0 \\ 0 & 0 \end{bmatrix} V^T x_n \right\|^2$

I - optimal Γ

optimal V : $\{v_j\}_{j=1}^{\tilde{d}}$ 'topmost' eigenvectors of $X^T X$

$\therefore$ optimal $\{w_j\} = \{v_j$ with $[\![r_j = 1]\!]\} =$ top eigenvectors

Radial Basis Function Network

linear aggregation of radial hypotheses

(1) radial : means only depends on distance between x and 'center' x_n

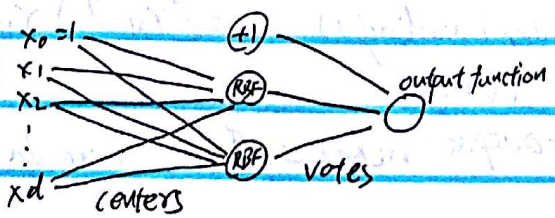
basis function : means to be 'combined'

let $g_n(x) = y_n \exp(-\gamma \|x - x_n\|^2)$:

$g_{svm}(x) = \text{sign}\left(\sum_{SV} \alpha_n g_n(x) + b \right)$

— linear aggregation of selected radial hypotheses

(2) RBF Network measure the similarity between input and centers μ_m



$h(x) = \text{Output}\left(\sum_{m=1}^{M} \beta_m \text{RBF}(x, \mu_m) + b \right)$

full RBF Network:

(3) $g_{uniform}(x) = sign\left(\sum_{m=1}^{\infty} y_m \exp(-\gamma\|x - x_m\|^2)\right)$

maximum when x closest to x_m, and the maximum one often dominate the $\sum_{m=1}^{M}$ term

$g_{nbor}(x) = $ ~~sign(~~ $\sum$ ~~)~~ y_m s.t. such that x closest to x_m

(4) if take squred error regression for output function
the whole network then turns to be a linear regression on RBF-transformed data: $z_n = [RBF(x_n, x_1), RBF(x_n, x_2) \cdots RBF(x_n, x_N)]$

optimal $\beta = (Z^T Z)^{-1} Z^T y = Z^{\dagger} y$

(5) make the number of centers fewer to be regularization
$\Rightarrow$ clustering with prototype $\Rightarrow$ k-means

① run K-means with $k = M$ to get $\{\mu_m\}$

② construct transform $\Phi(x)$ from RBF at μ_m

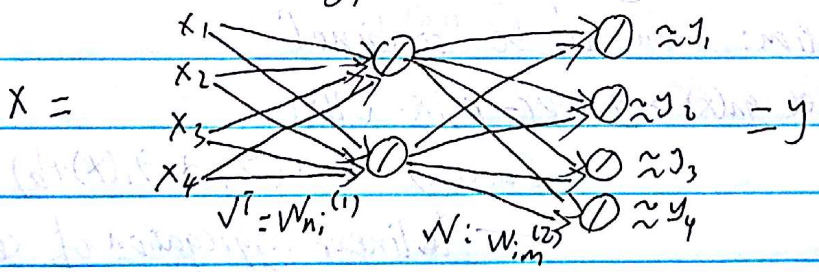
$\Phi(x) = [RBF(x, \mu_1), \cdots RBF(x, \mu_M)]$

③ run linear model on $\{(\Phi(x_n), y_n)\}$ to get β

④ return $g_{RBF-NET}(x) = $ LinearHypothesis $(\beta, \Phi(x))$

Matrix Factorization

(1) 'Linear Network' hypothesis



$\{(x_n = $ Binary Vector Encoding(n), $y_n = [r_{n1}\ ?\ ?\ r_{n4}\ r_{n5} \cdots r_{nM}]^T)\}$

hypothesis $h(x) = W^T V x$

per-user ~~oupt~~ output: $h(x_n) = W^T V_n$, where V_n is n-th column of V

(2) $E_{in}(\{w_m\},\{v_n\}) = \frac{1}{\sum_{m=1}^{M}|D_m|} \sum_{\substack{\text{user } n \\ \text{rated movie } m}} (r_{nm} - w_m^T v_n)^2$

Alternating Least Squares:

① initialize d dimension vectors $\{w_m\}, \{v_n\}$ (randomly initialized)

② alternating optimization of E_{in}: repeatedly

a. optimize $w_1, w_2, \ldots, w_M$: update w_m by m-th movie linear regression on $\{(v_n, r_{nm})\}$

b. optimize $v_1, v_2 \cdots v_N$: update v_n by n-th user linear regression on $\{(w_m, r_{nm})\}$

until converge.

(3) Gradient of Per-Example Error Function

$\nabla v_n = -2(r_{nm} - w_m^T v_n) w_m$

$\nabla w_m = -2(r_{nm} - w_m^T v_n) v_n$

SGD for Matrix Factorization

for $t = 0, 1 \ldots T$

① randomly pick (n, m) within all known r_{nm}

② calculate residual $\tilde{r}_{nm} = (r_{nm} - w_m^T v_n)$

③ SGD-update:

$v_n^{new} \leftarrow v_n^{old} + \eta \cdot \tilde{r}_{nm} w_m^{old}$

$w_m^{new} \leftarrow w_m^{old} + \eta \cdot \tilde{r}_{nm} v_n^{old}$

Finale

(1) Feature Exploitation Techniques: kernel, aggregation, extraction, low-dimensional

(2) Error Optimization Techniques: gradient, equivalence, stages

(3) Overfitting Elimination Techniques: regularization, validation