Chapter 3: The Linear model

A line is intuitively our first choice for a decision boundary.

Recap: perceptron learning algorithm:

We started by taking the hypothesis set that included all possible lines (actually hyperplanes) . The algorithm then searched for a good line in the hypothesis set by iteratively correcting the errors made by the current candidate line, in an attempt to improve Ein.

P43 Example 2.1: linear model: small VC dimension -> generalize well from Ein to Eout

3 important problems: Classification, regression(回归，复原), probability estimation

1. Linear model – linear classification：
   a) Recap:
      i. Generalize? (Eout ≈ Ein?):

Exercise 2.4

Exercise 2.4.

a). d_{VC}: for which $m_H = 2^N$, d_{VC} = the max N

x -> d+1 points

$$\begin{pmatrix} x_0 & \cdots & x_d \\ \vdots & & \vdots \\ x_0 & \cdots & x_d \end{pmatrix} \begin{pmatrix} w_0 \\ \vdots \\ w_d \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_d \end{pmatrix}$$

1个值

-> output

perceptron

d+1 dimension 里有d+1个坐标轴

∴ d+1个 ~~不相关~~ linearly independent 个值 / full rank

∴ ( ) -> non singular.

$Ax = B$ $\quad$ $x = A^{-1}B$

non singular ∴

∴ for every B (every result y), we can find x

$\begin{pmatrix} y_0 \\ \vdots \\ y_d \end{pmatrix}$ max 2^{d+1} 个结果

x: $\begin{pmatrix} w_0 \\ \vdots \\ w_d \end{pmatrix}$ the perceptron ~~[illegible]~~ Coefficient

∴ d_{VC} 至少 ≥ d+1

1个值, $m_H = 2^{d+1}$ 满足 shatter

Exercise 2.4 (b).

$$\begin{pmatrix} x_0 \\ \vdots \\ x_d \end{pmatrix}_1 \cdots \begin{pmatrix} x_0 \\ \vdots \\ x_d \end{pmatrix}_{d+2}$$

$$\therefore \begin{pmatrix} x_0 \\ \vdots \\ x_d \end{pmatrix}_{d+2} = \sum_{i=1}^{d+1} a_i \begin{pmatrix} x_0 \\ \vdots \\ x_d \end{pmatrix}_i$$

↳ linearly dependent

any d+2 vectors of d+1 $\overset{\text{length}}{\text{(dim)}}$ have to be linearly dependent

$$y_{d+2} = \sum_{i=1}^{d+1} a_i \, (x_0 \cdots x_d)_i \begin{pmatrix} w_0 \\ \vdots \\ w_d \end{pmatrix}$$

已知的.

$\therefore$ $y_1 \cdots y_{d+1} \longrightarrow$ determinant y_{d+2}.

$\therefore$ perceptron cannot shatter $d+2$.

$d_{vc} < d+2$. $d_{vc} \leq d+1$.

$\therefore d_{vc} = d+1$

of perceptron.

dvc of linear model = d+1

ii. VC generalization bound

$$E_{\text{out}}(g) \leq E_{\text{in}}(g) + \sqrt{\frac{8}{N} \ln \frac{4m_{\mathcal{H}}(2N)}{\delta}} \qquad (2.12)$$

2.10 - bound on the growth function in terms of dvc

$$m_{\mathcal{H}}(N) \leq N^{d_{\text{vc}}} + 1. \qquad (2.10)$$

we conclude that with high probability,

$$E_{\text{out}}(g) = E_{\text{in}}(g) + O\left(\sqrt{\frac{d}{N} \ln N}\right).$$

$$E_{out}(g) - E_{in}(g) \le \sqrt{\frac{8}{N} \ln \frac{4 m_{\mathcal{H}}(2N)}{\delta}}$$

$$\le \sqrt{\frac{8}{N} \ln \frac{4[(2N)^{d_{vc}}+1]}{\delta}}$$

提出 Constant term

$$\sqrt{\frac{\ln N^{d_{vc}}}{N}} \qquad d_{vc} = d+1$$

$$\sqrt{\frac{d \ln N}{N}}$$

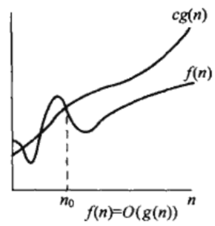
$O(\cdot)$ absolute value of this term is asymptotically smaller than a constant multiple of the argument

大O是我们在分析算法复杂度时最常用的一种表示法。

> f(x) = O(g(x)) 表示的含义是f(x)以g(x)为上界

当函数的大小只有上界，没有明确下界的时候，则可以使用大O表示法，该渐进描述符一般用于描述算法的 **最坏复杂度。**

f(x) = O(g(x))正式的数学定义：存在正常数c、n、n0，当 n>n0 的时，任意的 f(n) 符合 0 <= f(n) <= c.g(n)。如下图所示



$\therefore N \uparrow$, Eout $\approx$ Ein

b) Ein small? - Learn in sample a good hypothesis

Data is linearly separable: there is some hypothesis w* with Ein(w*) = 0.

PLA: increment based on one data point at a time; problem 1.3: finite step updating

any misclassified data point $(\mathbf{x}(t), y(t))$, and update $\mathbf{w}(t)$ as follows:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + y(t)\mathbf{x}(t).$$

PLA manages to search an infinite hypothesis set and output a linear separator in (provably) finite time.

Linearly separable data: from linearly separable target or not linearly separable target

But PLA converges & performance generalize well according to VC bound

2. Linear model – linear classification – non separable data

Cases:

1. Noisy or outlier;

2. Non-separable by a line

Both: PLA never terminates, jump from good to bad; No guarantee on Ein

Case1: tolerant small Ein;

Case2: nonlinear transformation

a) Case 1:

Hypothesis with min Ein

solve the combinatorial optimization problem:

$$\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \underbrace{\frac{1}{N}\sum_{n=1}^{N} [\![\operatorname{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n) \neq y_n]\!]}_{E_{\text{in}}(\mathbf{w})} .$$

NP-hard: no known efficient algorithm for it (NP 是指非确定性多项式（non-deterministic polynomial，缩写 NP）。所谓的非确定性是指，可用一定数量的运算去解决多项式时间内可解决的问题。)

∴approximately minimising Ein -> pocket algorithm: slower than PLA; no guarantee for convergence speed

The pocket algorithm:

1: Set the pocket weight vector $\hat{\mathbf{w}}$ to $\mathbf{w}(0)$ of PLA.
2: **for** $t = 0, \ldots, T \quad 1$ **do**
3: Run PLA for one update to obtain $\mathbf{w}(t+1)$.
4: Evaluate $E_{\text{in}}(\mathbf{w}(t+1))$.
5: If $\mathbf{w}(t+1)$ is better than $\hat{\mathbf{w}}$ in terms of E_{in}, set $\hat{\mathbf{w}}$ to $\mathbf{w}(t+1)$.
6: Return $\hat{\mathbf{w}}$.

b) Exercise 3.2 code explanation

[Learning-From-Data-A-Short-Course/Solutions to Chapter 3 The Linear Model.ipynb at master · niuers/Learning-From-Data-A-Short-Course (github.com)]

c) Example 3.1 (Handwritten digit recognition)

Raw input: preprocessed 16*16 pixel images

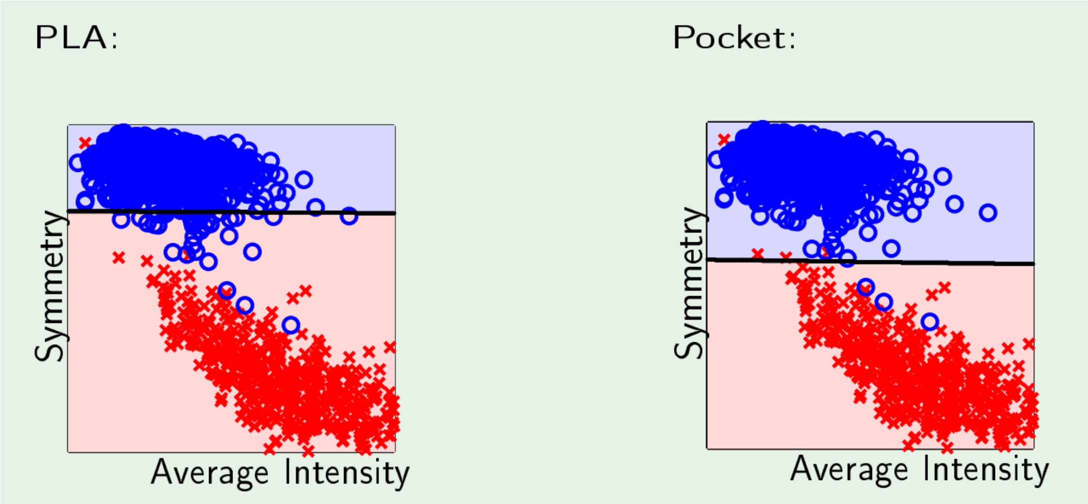
Decomposition: multiclass -> binary (classify {1,5} first)

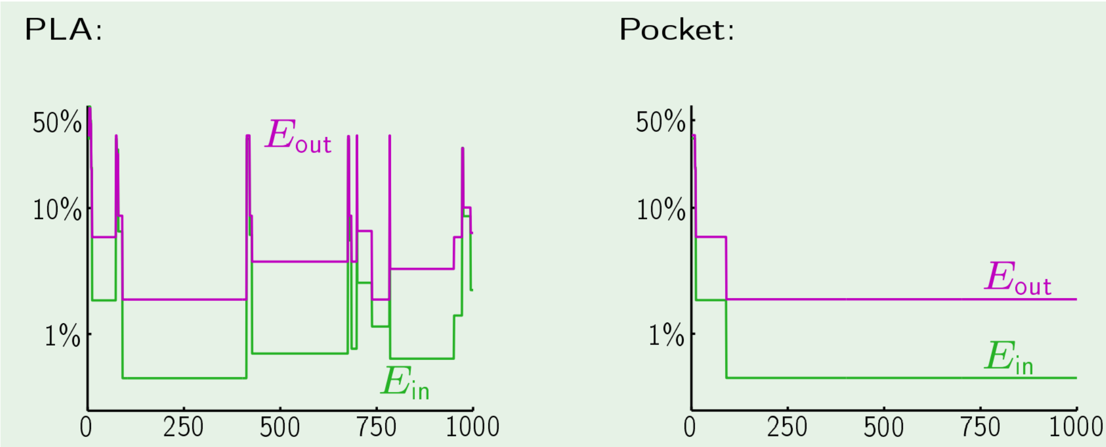
Processed input: average intensity & symmetry (asymmetry: absolute difference between an image and its FLIPPED versions; symmetry is the negation)

Situation: roughly separable by a line; poorly written digits prevent a perfect linear separation.

PLA: not stop updating; behavior unstable

Pocket algorithm: Ein decreasing





3. Linear model – linear regression
   a) A useful model applied to real-valued target function

Example: Set a credit limit for approved customers (assume linear seperable)

Input xn: customer info (eg. Historical records)

Output: Noisy target formalized as P(y|**x**) - expert different views and variations within them.

P(y|**x**) -> (**x**n,yn) -> g = argmin(Error between g(**x**) and y from P(y|**x**))

   b) The Algorithm

Minimizing the square error

$$E_{\text{out}}(h) = \mathbb{E}\left[(h(\mathbf{x}) - y)^2\right]$$

The Expectation uses joint probability distribution P(x, y) – unknown

Resort to Ein

$$E_{\text{in}}(h) = \frac{1}{N}\sum_{n=1}^{N}(h(\mathbf{x}_n) - y_n)^2$$

data matrix $X \in R^{N \times (d+1)}$

$$\text{N data Points}\left\{ \underbrace{\begin{pmatrix} x_0 & \cdots & x_d \\ \vdots & & \\ x_0 & \cdots & x_d \end{pmatrix}}_{d+1} \right. \begin{pmatrix} w_0 \\ \vdots \\ w_d \end{pmatrix} = \begin{pmatrix} h_1 \\ \vdots \\ h_N \end{pmatrix}$$

x, 1个data point

$x \in \{1\} \times R^d$

$x_0 = 1$

$h = Xw$

$$\therefore E_{in}(h) = \frac{1}{N} \sum_{n=1}^{N} (h(x_n) - y_n)^2$$

$$= \frac{1}{N} \sum_{n=1}^{N} (w^T x_n - y_n)^2$$

$$= \frac{1}{N} \| Xw - y \|^2$$

$$\begin{pmatrix} h_1 - y_1 \\ \vdots \\ h_N - y_N \end{pmatrix}$$ 它的 Euclidean norm^2

$$\Downarrow$$

$$= \frac{1}{N} (w^T X^T X w - 2 w^T X^T y + y^T y)$$

The linear regression algorithm is derived by minimizing Ein(w) over all possible $w \in R^{d+1}$

$$\mathbf{w}_{\text{lin}} = \underset{\mathbf{w} \in \mathbb{R}^{d+1}}{\operatorname{argmin}} E_{\text{in}}(\mathbf{w}).$$

find the w that minimizes Ein(w) by requiring that the **gradient of Ein with respect to w is the zero vector**

$$\nabla E_{\text{in}}(\mathbf{w}) = \mathbf{0}$$

$$[\nabla E_{\text{in}}(\mathbf{w})]_i = \frac{\partial}{\partial w_i} E_{\text{in}}(\mathbf{w}).$$

$$\nabla_w E_{in}(w) = \begin{pmatrix} \frac{\partial}{\partial w_0} E_{in}(w) \\ \frac{\partial}{\partial w_1} E_{in}(w) \\ \vdots \\ \frac{\partial}{\partial w_d} E_{in}(w) \end{pmatrix} = \frac{1}{N} \left((X^T X + X^T X) w - 2 X^T y \right)$$

$$= \frac{2}{N} (X^T X w - X^T y)$$

$(d+1) \times 1$

Rules: $$\nabla_{\mathbf{w}}(\mathbf{w}^{\mathrm{T}}\mathrm{A}\mathbf{w}) = (\mathrm{A}+\mathrm{A}^{\mathrm{T}})\mathbf{w}, \qquad \nabla_{\mathbf{w}}(\mathbf{w}^{\mathrm{T}}\mathbf{b}) = \mathbf{b}.$$

$$\mathrm{X}^{\mathrm{T}}\mathrm{X}\mathbf{w} = \mathrm{X}^{\mathrm{T}}\mathbf{y}.$$

If $\mathrm{X}^{\mathrm{T}}\mathrm{X}$ is invertible, $\mathbf{w} = \mathrm{X}^{\dagger}\mathbf{y}$ where $\mathrm{X}^{\dagger} = (\mathrm{X}^{\mathrm{T}}\mathrm{X})^{-1}\mathrm{X}^{\mathrm{T}}$ is the *pseudo-inverse*

If X^TX not invertible, pseudo-inverse can be defined, but solution not unique

Problem 3.15.

(a) $\operatorname{rank}(X) = \rho$ $\quad X \in R^{N\times(d+1)}$

$\underset{(d+1)\times N \;\; N\times(d+1)}{X^T X} \rightarrow (d+1)\times(d+1)$ No inverse

$\therefore \operatorname{rank}(X^TX) < d+1$

$\downarrow$

$\rho \le \min(d+1, N)$

$\downarrow$

$\rho \le d+1$

by the property of rank

$\operatorname{rank}(X) = \operatorname{rank}(X^TX) = \rho$

$\therefore \rho < d+1$

$$X = \underset{(N\times\rho)}{U}\,\overset{\rho\times\rho}{\Gamma}\,\underset{\rho\times(d+1)}{V^T}$$

(b) $w_{lin} = V\Gamma^{-1}U^Ty$

$$\begin{aligned} X^TXw_{lin} &= X^TXV\Gamma^{-1}U^Ty \\ &= (U\Gamma V^T)^T(U\Gamma V^T)V\Gamma^{-1}U^Ty \\ &= V\Gamma^TU^TU\Gamma V^TV\Gamma^{-1}U^Ty \\ &= V\Gamma^2\Gamma^{-1}U^Ty = V\Gamma U^Ty = X^Ty \end{aligned}$$

(c)

$$X^T X w = X^T y$$

$$X^T X w_{lin} = X^T y$$

$$X^T X (w - w_{lin}) = 0$$

$$V \Gamma U^T U \Gamma V^T (w - w_{lin}) = 0$$

$$V \Gamma^2 V^T (w - w_{lin}) = 0$$

$$\underbrace{\Gamma^{-2} V^T}_{\neq 0} \underbrace{V \Gamma^2 V^T (w - w_{lin})}_{=0} = V^T (w - w_{lin}) = 0$$

$\Gamma V^T \neq 0$ $\quad \Gamma \neq 0$

$\Gamma^{-1} \neq 0$

$\Gamma^{-1} V^T \neq 0$

$\Gamma^{-2} V^T \neq 0.$

let $w - w_{lin} = \delta$

$w_{lin} = w - \delta$ $\quad$ $w_{lin} \perp \delta$

orthogonal 证明:

$$w_{lin}^T \delta = w_{lin}^T (w - w_{lin})$$

$$= y^T U \Gamma^{-1} \underbrace{V^T (w - w_{lin})}_{=0}$$

$$= 0$$

$$w_{lin} = V \Gamma^{-1} U^T y$$

$A \to X^T X$, $z \to X^T y$

$$A w_{lin} = z$$

not full rank $\quad \rho < d+1$

$$A w_{lin} + \underset{=0}{AA'} = A(w_{lin} + A') = z$$

$(w_{lin} + A')$ 不唯一

存在 $A \perp A'$

$$\|w\|^2 = \|w_{lin} + \delta\|^2$$

$$= w_{lin}^T w_{lin} + 2 \underbrace{w_{lin}^T \delta}_{=0} + \delta^T \delta$$

$$= \|w_{lin}\|^2 + \|\delta\|^2$$

$$\geq \|w_{lin}\|^2$$

min norm

$X \in R^{N*(d+1)}$, N is much bigger than d+1, very likely there are d+1 linearly independent points
So rank($X^T X$) = d+1, invertible
Linear regression algorithm:

Linear regression algorithm:

1: Construct the matrix X and the vector $\mathbf{y}$ from the data set $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)$, where each $\mathbf{x}$ includes the $x_0 = 1$ bias coordinate, as follows

$$\underbrace{\mathrm{X} = \begin{bmatrix} \text{—}\mathbf{x}_1^T\text{—} \\ \text{—}\mathbf{x}_2^T\text{—} \\ \vdots \\ \text{—}\mathbf{x}_N^T\text{—} \end{bmatrix}}_{\text{input data matrix}}, \quad \underbrace{\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{\text{target vector}}.$$

2: Compute the pseudo-inverse $\mathrm{X}^\dagger$ of the matrix X. If $\mathrm{X}^T\mathrm{X}$ is invertible,

$$\mathrm{X}^\dagger = (\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T.$$

3: Return $\mathbf{w}_{lin} = \mathrm{X}^\dagger \mathbf{y}$.

Wlin has a decent Eout -> learning has occurred.
an analytic formula for learning -> widely used.

Estimated output: $\hat{\mathbf{y}} = \mathrm{X}\mathbf{w}_{\text{lin}}$ ---> $\hat{\mathbf{y}} = \mathrm{X}(\mathrm{X}^\intercal\mathrm{X})^{-1}\mathrm{X}^\intercal\mathbf{y}.$

linear transformation of the actual y through $\mathrm{H} = \mathrm{X}(\mathrm{X}^\intercal\mathrm{X})^{-1}\mathrm{X}^\intercal.$ (H 'puts a hat' on y)

c) Exercise 3.3

(a)
$$H^T = X\left[(X^TX)^{-1}\right]^T X^T$$
$$= X\left[(X^TX)^{-1}\right]X^T = H$$

(b)
$$H \times H = X(X^TX)^{-1}\underbrace{X^T X}(X^TX)^{-1}X^T$$
$$= X(X^TX)^{-1}X^T = H$$
$$\therefore H^k = H$$

(c)
$$(I-H)(I-H) = I - HI - IH + H^2$$
$$= I - H - H + H$$
$$= I - H$$
$$(I-H)^k = I - H$$

(d)
$$\underset{N\times N}{\mathrm{trace}(H)} = \mathrm{trace}(\underset{N\times d+1}{X}\,\underset{(d+1)\times(d+1)}{(X^TX)^{-1}}\,\underset{(d+1)\times N}{X^T}) = \mathrm{trace}(\underset{(d+1)N}{X^T}\,\underbrace{X(X^TX)^{-1}}_{N\times(d+1)})$$
$$\Downarrow \quad N\times N \qquad\qquad \Downarrow \quad (d+1)(d+1)$$
$$= \mathrm{trace}(I_{d+1}) = d+1$$

d) Generalization Issues

A regression version of the VC generalization bound

$$E_{\text{out}}(g) = E_{\text{in}}(g) + O\left(\frac{d}{N}\right)$$

Proof: Exercise 3.4

$$E_{\mathcal{D}}\left[E_{in}(w_{lin})\right] \quad = \sigma^2\left(1 - \frac{d+1}{N}\right)$$

$$E_{\mathcal{D},\epsilon'}\left[E_{test}(w_{lin})\right] \quad = \sigma^2\left(1 + \frac{d+1}{N}\right)$$

Exercise 3.4.

$$y = \underline{w}^{*T}\underline{x} + \varepsilon \qquad \underline{y} = \underline{X}w^{*} + \underline{\varepsilon}$$

(a)

Linear regression $\rightarrow$ $\hat{y} = Hy$

没有假设y是什么样的.

$$\hat{y} = Xw_{lin} = Hy$$

$$\begin{aligned}
\hat{y} &= Hy \\
&= H \cancel{w^{*T}x} (Xw^{*} + \varepsilon) \\
&= HXw^{*} + H\varepsilon \\
&= X(X^TX)^{-1}X^TXw^{*} + H\varepsilon \\
&= Xw^{*} + H\varepsilon
\end{aligned}$$

(b)

$$\begin{aligned}
\hat{y} - y &= Xw^{*} + H\varepsilon - Xw^{*} - \varepsilon = H\varepsilon - \varepsilon \\
&= (H - I)\varepsilon
\end{aligned}$$

(c)

$$\begin{aligned}
E_{in}(w_{lin}) &= \frac{1}{N}\sum_{n=1}^{n=N}\left(w_{lin}^T x_n - y_n\right)^2 \\
&= \frac{1}{N}\|Xw_{lin} - y\|^2 = \frac{1}{N}\|\hat{y} - y\|^2 \\
&= \frac{1}{N}\|(H-I)\varepsilon\|^2 = \frac{1}{N}\varepsilon^T(H-I)^T(H-I)\varepsilon \\
&= \frac{1}{N}\varepsilon^T(H^T-I)(H-I)\varepsilon \\
&= \frac{1}{N}\varepsilon^T(I-H)\varepsilon
\end{aligned}$$

Exercise 3.3 (c)

$\overset{N\times N}{H} = X(X^T X)^{-1} X^T$

$\overset{1\times N}{(\varepsilon_1 \cdots \varepsilon_N)} \overset{N\times N}{\begin{pmatrix} H_{11} & \cdots \\ \vdots & \\ H_{N1} & \cdots \end{pmatrix}} \underset{(N\times 1)}{\begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_N \end{pmatrix}}$

$\overset{(1\times N)}{(\cdot \cdot \cdots \cdot)}$

(d)

$$E_D[E_{in}(w_{lin})] = E_D\left[\frac{1}{N}\varepsilon^T(I-H)\varepsilon\right]$$

$$= \frac{1}{N}\left[E_D(\varepsilon^T\varepsilon) - E_D(\varepsilon^T H \varepsilon)\right]$$

$$= \frac{1}{N}\left[E_D\left(\sum_{n=1}^{n=N}\varepsilon_n^2\right) - E_D\left(\sum_{j=1}^{j=N}\sum_{i=1}^{i=N}\varepsilon_i h_{ij}\,\varepsilon_j\right)\right]$$

$$= \frac{1}{N}\left[\sum_{n=1}^{n=N}E_D(\varepsilon_n^2) - \sum_{j=1}^{j=N}\sum_{i=1}^{i=N}E_D(\varepsilon_i h_{ij}\varepsilon_j)\right]$$

$$= \frac{1}{N}\left[N\sigma^2 - \sum_{i=1}^{i=N}E_D(\varepsilon_i^2 h_{ii})\right]$$

$$= \frac{1}{N}\left[N\sigma^2 - \sum_{i=1}^{i=N}E_D(\varepsilon_i^2)E_D(h_{ii})\right]$$

$$= \frac{1}{N}\left[N\sigma^2 - \sigma^2\sum_{i=1}^{i=N}E_D(h_{ii})\right]$$

$$= \frac{1}{N}\left[N\sigma^2 - \sigma^2 E_D(\mathrm{trace}(H))\right)$$

$$= \frac{1}{N}\left[N\sigma^2 - \sigma^2 \mathrm{trace}(H)\right)$$

$$\underbrace{E_D(\varepsilon_i h_{ij}\varepsilon_j)}_{\text{independent if } i\neq j} = \underset{\overset{\shortparallel}{0}}{E_D(\varepsilon_i)}\, E_D(h_{ij}\varepsilon_j) = 0$$

$$= \frac{1}{N}\left[N\sigma^2 - \sigma^2(d+1)\right]$$

$$= \sigma^2 - \frac{\sigma^2(d+1)}{N}$$

$$= \sigma^2\left(1 - \frac{d+1}{N}\right)$$

(e)

$$E_{D,\epsilon'}[E_{test}(w_{lin})] = E_{D,\epsilon'}\left[\frac{1}{N}\|\hat{y}-y'\|^2\right]$$

$$= E_{D,\epsilon'}\left[\frac{1}{N}\|\underbrace{Hy-y'}_{\substack{Xw^* + H\epsilon - (Xw^* + \epsilon') \\ \downarrow \\ H\epsilon - \epsilon'}}\|^2\right]$$

y' from D_{test}
y from D_{train}
share same x_n
but different ϵ

$$= E_{D,\epsilon'}\left[\frac{1}{N}\|H\epsilon-\epsilon'\|^2\right]$$

$$= E_{D,\epsilon'}\left[\frac{1}{N}(H\epsilon-\epsilon')^T(H\epsilon-\epsilon')\right]$$

$$= E_{D,\epsilon'}\left[\frac{1}{N}(\epsilon^T H - \epsilon'^T)(H\epsilon-\epsilon')\right]$$

$$= E_{D,\epsilon'}\left[\frac{1}{N}(\epsilon^T H H \epsilon - \epsilon'^T H\epsilon - \epsilon^T H\epsilon' + \epsilon'^T\epsilon)\right]$$

$$= E_{D,\epsilon'}\left[\frac{1}{N}(\epsilon^T H \epsilon - \underbrace{\epsilon'^T H\epsilon - \epsilon^T H\epsilon'}_{一样} + \epsilon'^T\epsilon')\right]$$

$$= E_{D,\epsilon'}\left[\frac{1}{N}(\epsilon^T H \epsilon + \epsilon'^T\epsilon)\right]$$

$$= \frac{1}{N}\left\{\sum_{i=1}^{i=N}\sum_{j=1}^{j=N} E_{D,\epsilon'}[\epsilon_i h_{ij}\epsilon_j] + \sum_{i=1}^{i=N} E_{D,\epsilon'}[\epsilon_i'^2]\right\}$$

$$= \frac{1}{N}\left\{\sigma^2(d+1) + N\sigma^2\right\} = \sigma^2\left(1+\frac{d+1}{N}\right)$$

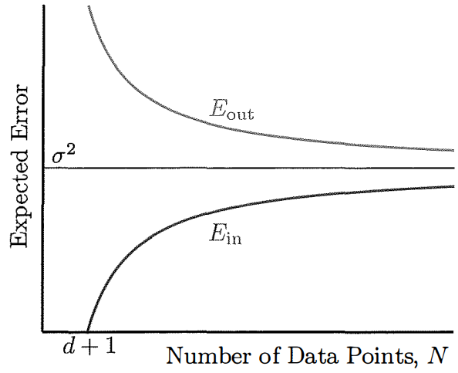


Figure 3.4: The learning curve for linear regression.

Ein: learned linear fit has degree of freedom d+1 (∴ Ein 可以= 0 when d+1 points) -> eat into in-sample noise (∵ cannot distinguish)

Eout: best with σ^2, additional error due to fitting in the in-sample noise

Recap:

Linear model - 'draw a line' betwen 2 categories

1)linear classification

2)linear regression

linear classification - perceptron learning algorithm (PLA)

as long as data linearly seperable, Ein = 0

what is non linearly seperable?

-two cases:

1. linearly seperable after discarding a few examples
2. not linearly seperable, but sperable by a more sophisticated curve

case 1 - The pocket algorithm

- put the best in the pocket
- get the best of the random selected lines

An example - handwritten digit recognition

1. multiclass diciomposition: recognise 1 / 5
2. 256 pixels -> 2 features (symmetry and intensity)

test: use pocket, Eout lower than human error

2) linear regression:

target function real-valued, not y generated from p(y|x)

what is the Ein and Eout?

The hypothesis h(x) is still a linear combination of the compontents of x

Now we want to minimise Ein

Ein(w) differentiable -> differentiation wrt w,

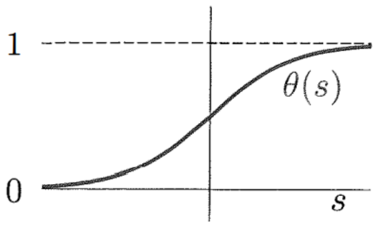
4. The linear model – Logistic Regression

Output a probability between 0 and 1: real (like linear regression) and bounded (like linear classification). Eg. Occurrence of heart attacks

a) The linear model – Logistic Regression (model) - Predicting a probability

$$h(\mathbf{x}) = \theta(\mathbf{w}^{\mathrm{T}}\mathbf{x}),$$

where θ is the so-called *logistic* function $\theta(s) = \frac{e^s}{1+e^s}$ whose output is between 0 and 1.



Output: probability of a binary event – uncertain classification

i. Exercise 3.5

Let $s \to \infty$, we have $e^{-2s} \to 0$, thus $\tanh(s) \to 1$. Similarly, when $s \to -\infty$, $\tanh(s) \to -1$.

It's also easy to see that $\tanh(s) < 1$ and $\tanh(s) > -1$. So -1 and 1 are hard thresholds for $\tanh(s)$ when $|s|$ is large.

When $|s|$ is small, consider Taylor expansion of $e^s = 1 + s + \frac{s^2}{2} + O(s^3)$, then $e^{-s} = 1 - s + \frac{s^2}{2} - O(s^3)$, we have $\tanh(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}} = \frac{2s + 2O(s^3)}{2+s^2} \approx s$.

So $\tanh(s)$ is approximately linear when $|s|$ is small. There's no threshold in this case.

Exersize 3.5

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

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

$\tanh(s) + 1 = 2\theta(2s)$

$\tanh(s) = 2\theta(2s) - 1$

(b) $s \to \infty$ $\quad \tanh(s) \to \frac{e^s - 0}{e^s + 0} = 1$ $\triangle$

$s \to -\infty$ $\quad \frac{0 - e^{-s}}{0 + e^{-s}} = -1$ $\triangle$

$\cancel{s \to 0 \quad \tanh(s) \to}$

$e^s = 1 + s + \frac{s^2}{2} + O(s^3)$

$e^{-s} = 1 - s + \frac{s^2}{2} - O(s^3)$

$\tanh = \frac{2s + \cancel{2O(s^3)}}{2 + \cancel{s^2}} \approx s$ $\quad$ linear

$$P(y \mid \mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{for } y = +1; \\ 1 - f(\mathbf{x}) & \text{for } y = -1. \end{cases} \qquad f(\mathbf{x}) = \mathbb{P}[y = +1 \mid \mathbf{x}].$$

the data is in fact generated by a noisy target $P(y \mid \mathbf{x})$,

Error measure: how close h to f in terms of the noisy ±1 examples.
推导出 error measure：

$$P(y \mid \mathbf{x}) = \begin{cases} h(\mathbf{x}) & \text{for } y = +1; \\ 1 - h(\mathbf{x}) & \text{for } y = -1. \end{cases}$$

We substitute for $h(\mathbf{x})$ by its value $\theta(\mathbf{w}^{\mathsf{T}}\mathbf{x})$, and use the fact that $1 - \theta(s) = \theta(-s)$ (easy to verify) to get

$$P(y \mid \mathbf{x}) = \theta(y\ \mathbf{w}^{\mathsf{T}}\mathbf{x}). \tag{3.8}$$

the probability of :
given x, get the actual y (generated by the noisy target P(y|x))
data points in a set: independent: the probability of getting all the actual y from all the given x:

$$\prod_{n=1}^{N} P(y_n \mid \mathbf{x}_n).$$

Method of maximum likelihood: maximize $\prod_{n=1}^{N} P(y_n \mid \mathbf{x}_n).$

Equivalent to minimize:

$$-\frac{1}{N}\ln\left(\prod_{n=1}^{N} P(y_n \mid \mathbf{x}_n)\right) = \frac{1}{N}\sum_{n=1}^{N}\ln\left(\frac{1}{P(y_n \mid \mathbf{x}_n)}\right) = \frac{1}{N}\sum_{n=1}^{N}\ln\left(\frac{1}{\theta(y_n\mathbf{w}^{\mathsf{T}}\mathbf{x}_n)}\right)$$

minimize the “error” with respect to the weight vector w

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\ln\left(1 + e^{-y_n\mathbf{w}^{\mathsf{T}}\mathbf{x}_n}\right).$$

Pointwise error measure

$$\mathrm{e}(h(\mathbf{x}_n), y_n) = \ln(1+e^{-y_n\mathbf{w}^{\mathsf{T}}\mathbf{x}_n})$$

error is small when actual output (y_n) and the estimation ($w^{T}x_n$) are consistent (same sign)
error measure encourages w to 'classify' each Xn correctly

ii. Exercise 3.6 (prove error measure)

Exercise 3.6

1. (a) The probability to get y_n is $P(y_n|x_n)$, by maximum likelihood method, we need maximize the likelihood, $\prod_{n=1}^{N} P(y_n|x_n)$, this is equivalent to maximize the logrithm of it: $\sum_{n=1}^{N} \ln(P(y_n|x_n))$, or minimize the negative of it: $-\sum_{n=1}^{N} \ln(P(y_n|x_n))$

When $y_n = +1$, $P(y_n|x_n) = h(x_n)$, and when $y_n = -1$, $P(y_n|x_n) = 1 - h(x_n)$, separate the cases for $y_n = 1$ and $y_n = -1$, we have:

$$\begin{aligned}
E_{in}(w) &= -\sum_{n=1}^{N} \ln(P(y_n|x_n)) \\
&= -\sum_{n=1}^{N} I(y_n = +1)\ln h(x_n) + I(y_n = -1)\ln(1 - h(x_n)) \\
&= \sum_{n=1}^{N} I(y_n = +1)\ln\frac{1}{h(x_n)} + I(y_n = -1)\ln\frac{1}{(1 - h(x_n))}
\end{aligned}$$

1. (b) For $h(x) = \theta(w^T x) = \frac{e^{w^T x}}{1+e^{w^T x}}$, we have $\ln\frac{1}{h(x_n)} = \ln(1 + e^{-w^T x_n})$ and $\ln\frac{1}{(1-h(x_n))} = \ln(1 + e^{w^T x_n})$. Combine them together we have

$$\begin{aligned}
E_{in}(w) &= \sum_{n=1}^{N} I(y_n = +1)\ln(1 + e^{-w^T x_n}) + I(y_n = -1)\ln(1 + e^{w^T x_n}) \\
&= \sum_{n=1}^{N} \ln(1 + e^{-y_n w^T x_n})
\end{aligned}$$

Which is equivalent of minimizing the one in equation (3.9).

这种 error measure 是 cross-entropy error measure

$$p\log\frac{1}{q} + (1-p)\log\frac{1}{1-q}.$$

iii. Exercise 3.7

Exercise 3.7

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

$$\nabla E_{in}(w) = \frac{1}{N}\sum_{n=1}^{N} \frac{-y_n x_n e^{-y_n w^T x_n}}{1+e^{-y_n w^T x_n}} \rightarrow \frac{\partial}{\partial w_i} \begin{pmatrix} \vdots \end{pmatrix} \text{ column vector.}$$

$$= -\frac{1}{N}\sum_{n=1}^{N} y_n x_n \frac{1}{1+e^{y_n w^T x_n}} \qquad \theta(s) = \frac{e^s}{1+e^s}$$

$$1-\theta(s) = \theta(-s) \downarrow \frac{1}{1+e^s}$$

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

if missclassified. $y_n w^T x_n < 0$

$\theta(-y_n w^T x_n) > 0.5 \rightarrow$ contribute more to $\nabla E_{in}(w)$

correctly classified $\theta(-y_n w^T x_n) < 0.5$

Iteratively set the gradient of the error measure to zero – gradient descent

b) The linear model – Logistic Regression (model) – Gradient descent

Train learning models with smooth error measures – minimizing a twice-differentiable function

not necessarily rest in the global minima - end up at a local minimum (depend on starting weights)

but logistic regression with the cross-entropy error – only one (unique) global minimum – Ein(w) convex

i. take a step in the direction of steepest descent 推导:

Taylor expansion:

$$\begin{aligned} \Delta E_{\text{in}} &= E_{\text{in}}(\mathbf{w}(0)+\eta\hat{\mathbf{v}}) - E_{\text{in}}(\mathbf{w}(0)) \\ &= \eta \nabla E_{\text{in}}(\mathbf{w}(0))^{\text{T}}\hat{\mathbf{v}} + O(\eta^2) \\ &\geq \eta \|\nabla E_{\text{in}}(\mathbf{w}(0))\|, \end{aligned}$$

For largest negative ΔEin = $\eta\|\nabla E_{\text{in}}(\mathbf{w}(0))\|$ (ignore square term)

$$\hat{\mathbf{v}} = -\frac{\nabla E_{\text{in}}(\mathbf{w}(0))}{\|\nabla E_{\text{in}}(\mathbf{w}(0))\|}.$$

leads to the largest decrease in Ein

Exercise 3.8

$\hat{v}$ is the direction which gives largest decrease in E_{in} only holds for small η, that's because when η is large, we can't ignore the squared term and smaller terms in the Taylor expansion. The lower bound can't be achieved.

ii. Learning rate η:

Fixed too small step size: insufficient when far from local minimum

Too large step size: bouncing around, possibly increasing Ein

Variable η: large step when far from the minimum

$$\eta_t = \eta \|\nabla E_{\text{in}}\|$$

$$W(t+1) = W(t) - \underbrace{\eta \|\nabla E_{in}\|}_{\text{Variable } \eta_t} \frac{\nabla E_{in}(W(t))}{\|\nabla E_{in}(W(t))\|}$$

$$= W(t) - \underbrace{\eta}_{\text{fixed } \eta_t} \nabla E_{in}(W(t))$$

Fixed learning rate gradient descent:

Initialize the weights at time step $t = 0$ to $\mathbf{w}(0)$.
for $t = 0, 1, 2, \ldots$ **do**
  Compute the gradient $\mathbf{g}_t = \nabla E_{\text{in}}(\mathbf{w}(t))$.
  Set the direction to move, $\mathbf{v}_t = -\mathbf{g}_t$.
  Update the weights: $\mathbf{w}(t+1) = \mathbf{w}(t) + \eta \mathbf{v}_t$.
  Iterate to the next step until it is time to stop.
Return the final weights.

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \ln\left(1 + e^{-y_n \mathbf{w}^{\mathrm{T}} \mathbf{x}_n}\right).$$

Logistic regression algorithm:

Initialize the weights at time step $t = 0$ to $\mathbf{w}(0)$.
for $t = 0, 1, 2, \ldots$ **do**
  Compute the gradient

$$\mathbf{g}_t = -\frac{1}{N} \sum_{n=1}^{N} \frac{y_n \mathbf{x}_n}{1 + e^{y_n \mathbf{w}^{\mathrm{T}}(t) \mathbf{x}_n}}.$$

  Set the direction to move, $\mathbf{v}_t = -\mathbf{g}_t$.
  Update the weights: $\mathbf{w}(t+1) = \mathbf{w}(t) + \eta \mathbf{v}_t$.
  Iterate to the next step until it is time to stop.
Return the final weights $\mathbf{w}$.

iii. Initialization and termination

Initialization:

w(0) = **0** - stuck on a perfectly symmetric hilltop.

safer: randomly - Choosing each weight independently from a Normal distribution with zero mean and small variance

Termination:

1. set an upper bound on the number of iterations (no guarantee on the quality)

2. (∵at minimum, gradient = 0) gradient below a certain threshold (if flat region, prematurely stop)
3. Require error itself small

Combination: a maximum number of iterations, marginal error improvement, coupled with small value for the error itself