CS 480 Personal Notes

Marcus Chan

Taught by Hongyang Zhang

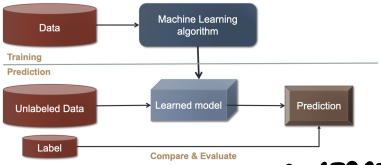
UW CS '25

Chapter 1: Perceptrons

ML

"Machine learning" is a branch of AI that focuses on methods that learn from data & make predictions on unseen data.

3 phases:
1. training;
2. prediction; &
3. evaluation.



PARADIGMS OF ML ALGOS (TRAINING)

1. "Supervised model": learning with labelled data (x, y)
   eg email classification, image classification
2. "Unsupervised model": discover patterns in unlabeled data x
   eg cluster similar data points, reduce data dimension etc
3. "Semi-supervised model": using both labelled & unlabelled data

WHAT A DATASET LOOKS LIKE

	Training samples						Test samples	
	x_1	x_2	x_3	x_4	$\cdots$	x_n	x_1'	x_2'
$\mathbb{R}^d \ni$ Feature	0	1	0	1	$\cdots$	1	1	0.9
	0	0	1	1	$\cdots$	0	1	1.1
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
	1	0	1	0	$\cdots$	1	1	-0.1
Label y	$+$	$+$	$-$	$+$	$\cdots$	$-$	?	?

- each column is a data point, n in total & each with d features
- y is the "label vector"
- x_1' & x_2' are the test samples whose labels need to be predicted.
  (we use "x'" to denote test samples)

INNER PRODUCT: $\langle x, w \rangle$

Define the "inner product" of a & b to be

$$\langle a, b \rangle = \sum_j a_j b_j,$$

where a_j, b_j are the j^{th} entries of a & b.

LINEAR FUNCTION

We say a function f is "linear" if

$$f(\alpha x + \beta z) = \alpha f(x) + \beta f(z) \quad \forall \alpha, \beta \in \mathbb{R}, \; x, z \in \mathbb{R}^d.$$

Equivalently, f is linear iff there exists $w \in \mathbb{R}^d$ such that

$$f(x) = \langle x, w \rangle = \sum_j x_j w_j.$$

Proof. ($\Rightarrow$) Let $w = [f(e_1), \ldots, f(e_d)]$, where e_i is the i^{th} coordinate vector. Then

$$\begin{aligned} f(x) &= f(x_1 e_1 + \cdots + x_d e_d) \\ &= x_1 f(e_1) + \cdots + x_d f(e_d) \\ &= \langle x, w \rangle. \end{aligned}$$

($\Leftarrow$) Note

$$\begin{aligned} f(\alpha x + \beta z) &= \langle \alpha x + \beta z, w \rangle \\ &= \alpha \langle x, w \rangle + \beta \langle z, w \rangle \\ &= \alpha f(x) + \beta f(z). \end{aligned}$$

∎

AFFINE FUNCTION

We say f is an "affine function" if there exists a $w \in \mathbb{R}^d$, $b \in \mathbb{R}$ such that

$$f(x) = \langle x, w \rangle + b \quad \forall x \in \mathbb{R}^d.$$

SCORE: $\hat{y}$

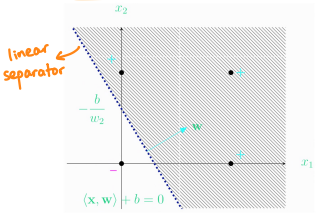
Given $w \in \mathbb{R}^d$, $b \in \mathbb{R}$, define the "score" at some $x \in \mathbb{R}^d$ to be

$$\text{Score}(x) = \langle x, w \rangle + b.$$

Our "prediction" for y is then

$$\hat{y} = \text{sign}(\text{score}(x)) = \begin{cases} +1, & \text{score}(x) > 0 \\ -1, & \text{score}(x) \le 0. \end{cases}$$

We want to tune w, b so that "$\hat{y} = y$" for each x.



- x is free, w & b fixed
- w & b uniquely determine the linear separator.

PERCEPTRONS

Algorithm for training:

Algorithm 1 Training Perceptron

Input: Dataset $= (\mathbf{x}_i, \mathrm{y}_i) \in \mathbb{R}^d \times \{\pm 1\} : i = 1, \ldots, n$, initialization $\mathbf{w}_0 \in \mathbb{R}^d$ and $b_0 \in \mathbb{R}$
Output: $\mathbf{w}$ and b (so a linear classifier $\mathrm{sign}(\langle \mathbf{x}, \mathbf{w} \rangle + b)$)
for $t = 1, 2, \ldots$ **do**
  receive index $I_t \in \{1, \ldots, n\}$ // I_t can be random
  if $\mathrm{y}_{I_t}(\langle \mathbf{x}_{I_t}, \mathbf{w} \rangle + b) \leq 0$ // a "mistake" happens
  then
    $\mathbf{w} \leftarrow \mathbf{w} + \mathrm{y}_{I_t}\mathbf{x}_{I_t}$
    $b \leftarrow b + \mathrm{y}_{I_t}$ // update after a "mistake"
  end
end

note: we can just set $I_t = t$

- we typically set $w_0 = 0$ & $b_0 = 0$
- we only update after a mistake (aka "lazy update")
- note we are going through the data one by one.

In particular, we want to find $w \in \mathbb{R}^d$, $b \in \mathbb{R}$ such that for all $i = 1, \ldots, n$,

$$y_i(\langle x_i, w \rangle + b) > 0.$$

Note that if a mistake happens on (x, y):

$$\begin{aligned} y[\langle x, w_{k+1} \rangle + b_{k+1}] &= y[\langle x, w_k + yx \rangle + b_k + y] \\ &= y[\langle x, w_k \rangle + y\langle x, x \rangle + b_k + y] \\ &= y[\langle x, w_k \rangle + y\|x\|_2^2 + b_k + y] \\ &= y[\langle x, w_k \rangle + b_k] + y^2\|x\|_2^2 + y^2 \\ &= y[\langle x, w_k \rangle + b_k] + \underbrace{\|x\|_2^2 + 1}_{\text{always positive \& } \geq 1.} \quad \because y = \pm 1 \end{aligned}$$

Example: spam filtering.

	x_1	x_2	x_3	x_4	x_5	x_6
and	1	0	0	1	1	1
viagra	1	0	1	0	0	0
the	0	1	1	0	1	1
of	1	1	0	1	0	1
nigeria	1	0	0	0	1	0
y	+	−	+	−	+	−

- Recall the update: $\mathbf{w} \leftarrow \mathbf{w} + \mathrm{y}\mathbf{x}, \quad b \leftarrow b + \mathrm{y}$ (when a mistake happens on $(\mathbf{x}, \mathrm{y})$)
  - $\mathbf{w}_0 = [0, 0, 0, 0, 0, 0], \quad b_0 = 0 \implies \mathrm{score}(\mathbf{x}_1) = 0 \implies \hat{\mathrm{y}}_1 = - \quad$ x
  - $\mathbf{w}_1 = [1, 1, 0, 1, 1, 1], \quad b_1 = 1 \implies \mathrm{score}(\mathbf{x}_2) = 2 \implies \hat{\mathrm{y}}_2 = + \quad$ x
  - $\mathbf{w}_2 = [1, 1, -1, 0, 1], b_2 = 0 \implies \mathrm{score}(\mathbf{x}_3) = 0 \implies \hat{\mathrm{y}}_3 = - \quad$ x
  - $\mathbf{w}_3 = [1, 2, 0, 0, 1], \quad b_3 = 1 \implies \mathrm{score}(\mathbf{x}_4) = 2 \implies \hat{\mathrm{y}}_4 = + \quad$ x
  - $\mathbf{w}_4 = [0, 2, 0, -1, 1], b_4 = 0 \implies \mathrm{score}(\mathbf{x}_5) = 1 \implies \hat{\mathrm{y}}_5 = + \quad$ ✓
  - $\mathbf{w}_4 = [0, 2, 0, -1, 1], b_4 = 0 \implies \mathrm{score}(\mathbf{x}_6) = -1 \implies \hat{\mathrm{y}}_6 = - \quad$ ✓

A TRICK TO HIDE THE BIAS TERM

Note that

$$\langle x, w \rangle + b = \left\langle \begin{pmatrix} x \\ 1 \end{pmatrix}, \begin{pmatrix} w \\ b \end{pmatrix} \right\rangle$$

(x_{pad}, w_{pad})

This is a "trick" to ignore b in future calculations.

Thus, our new update rule is

$$w_{pad} \leftarrow w_{pad} + y x_{pad}.$$

CONVERGENCE THEOREM (LINEARLY SEPARABLE CASE)

Suppose there exists a w^* such that

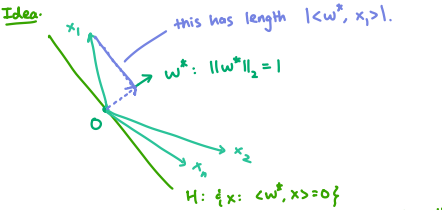
$$y_i \langle x_i, w^* \rangle > 0 \quad \forall i = 1, \dots, n.$$

Assume $\|x_i\|_2 \leq C \ \forall i$ and that w^* is normalized so that $\|w^*\|_2 = 1$.

Define the margin $\gamma := \min_i |\langle x_i, w^* \rangle|$.

Then the Perceptron algorithm converges after C^2/γ^2 mistakes.

Idea:



- w^* is our "perfect" solution for w (ie the "goal" criteria is satisfied).
- thus, we want to show w "converges" to w^*.

Proof. Recall the update is $w \leftarrow w + yx$.

Define

$$\cos(w, w^*) = \frac{\langle w, w^* \rangle}{\|w\| \, \|w^*\|} = \frac{\langle w, w^* \rangle}{\|w\|}$$

(since we defined $\|w^*\| = 1$).

Consider an update and its effect on $\langle w, w^* \rangle$:

$$\langle w, w^* \rangle \longrightarrow \langle w + yx, w^* \rangle$$

$$= \langle w, w^* \rangle + \underbrace{y \langle x, w^* \rangle}_{\text{positive } \because w^* \text{ is perfect}}$$

$$= \langle w, w^* \rangle + |\langle x, w^* \rangle|$$

$$\geq \langle w, w^* \rangle + \gamma.$$

This means for each update, $\langle w, w^* \rangle$ grows by at least $\gamma > 0$.

Similarly, consider an updates effect on $\|w\|_2^2$:

$$\|w\|_2^2 = \langle w, w \rangle \longrightarrow \langle w + yx, w + yx \rangle$$

$$= \langle w, w \rangle + \underbrace{2y \langle x, x \rangle}_{< 0} + y^2 \langle x, x \rangle$$

$$= \langle w, w \rangle + 2y \langle w, x \rangle + \underbrace{\|x\|_2^2}_{\leq C^2}$$

$$\leq \langle w, w \rangle + C^2.$$

This means for each update, $\langle w, w \rangle$ grows by at most C^2.

Now, let $w_0 = 0$. We now know after M updates:

$$\langle w_M, w^* \rangle \geq \langle w_{M-1}, w^* \rangle + \gamma$$

$$\geq \langle w_{M-2}, w^* \rangle + 2\gamma$$

$$\geq \dots \geq \underbrace{\langle w_0, w^* \rangle}_{=0} + M\gamma$$

$$= M\gamma.$$

Similarly, note

$$\langle w_M, w_M \rangle \leq \langle w_{M-1}, w_{M-1} \rangle + C^2$$

$$\leq \dots \leq \underbrace{\langle w_0, w_0 \rangle}_{=0} + MC^2$$

$$\leq MC^2.$$

Since

$$\cos(w, w^*) = \frac{\langle w, w^* \rangle}{\|w\|} \leq 1 \implies \langle w, w^* \rangle \leq \|w\|$$

Therefore

$$M\gamma \leq \langle w, w^* \rangle \leq \|w\| \leq \sqrt{MC^2} = \sqrt{M}\, C.$$

Rearranging, this tells us that $M \leq \frac{C^2}{\gamma^2}$, which finishes the proof. ☑

In particular, the larger γ is, the more separable the data is, and hence the faster the algorithm converges!

ANOTHER PERSPECTIVE ON PERCEPTRONS

1. Our hypothesis is $\hat{y} = \text{sign}\{\langle w, x\rangle\}$.

2. We can define our "loss function" as

$$\ell(w; x_t, y_t) = -y_t \langle w, x_t\rangle \mathbb{I}(\text{mistake on } (x_t, y_t))$$
$$= \begin{cases} -y_t\langle w, x_t\rangle, & \text{if mistake happens} \iff y_t\langle w, x_t\rangle < 0 \\ 0, & \text{otherwise} \end{cases}$$
$$= -\min\{0,\ y_t\langle w, x_t\rangle\}.$$

3. The average of all the loss functions of the data points is then

$$L(w) = -\frac{1}{n}\sum_{t=1}^{n} y_t \langle w, x_t\rangle \mathbb{I}[\text{mistake on } x_t].$$

4. Our gradient descent update:

$$w_{t+1} = w_t - \eta_t \nabla_w \ell(w_t, x_t, y_t) = w_t + \eta_t y_t x_t \mathbb{I}[\text{mistake on } x_t].$$

If we set the step size $\eta_t = 1$, then

$$w_{t+1} = w_t + y_t x_t,$$

which is our update rule.

PERCEPTRONS ARE NOT UNIQUE

Note perceptrons are not unique as the algorithm terminates as long as there is no mistake.

- it depends on initialization & our sampling rule of I_t.

MAXIMIZE MARGIN

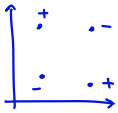
We want to choose w such that

$$w = \max_{w:\ \forall i,\ \hat{y}_i y_i > 0} \ \min_{i=1,\dots,n} \frac{\hat{y}_i y_i}{\|w\|}, \qquad \hat{y}_i := \langle x_i, w\rangle + b.$$

XOR DATASET

There is no line that can separate + from −.

x_1	x_2	x_3	x_4
0	1	0	1
0	0	1	1
−	+	+	−



What if we run Perceptron?

Suppose $\exists w, b$ s.t. $y(\langle x, w\rangle + b) > 0$.

Then:

$x_1 = (0,0)$, $y_1 = -$ $\Rightarrow$ $b < 0$

$x_2 = (1,0)$, $y_2 = +$ $\Rightarrow$ $w_1 + b > 0$

$x_3 = (0,1)$, $y_3 = +$ $\Rightarrow$ $w_2 + b > 0$

} $w_1 + w_2 + 2b > 0$

$x_4 = (1,1)$, $y_4 = -$ $\Rightarrow$ $w_1 + w_2 + 2b < 0$.

Hence

$$\underbrace{(w_1 + w_2 + 2b)}_{>0} - \underbrace{(w_1 + w_2 + b)}_{<0} = b > 0,$$

which contradicts our earlier statement that $b < 0$.

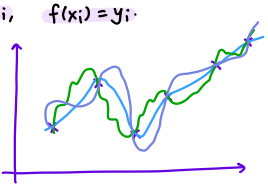
HARDNESS RESULT (NON-LINEARLY SEPARABLE CASE)

If there is no perfect separating hyperplane for our data, then the Perceptron algorithm cycles.

Chapter 2: Linear Regression

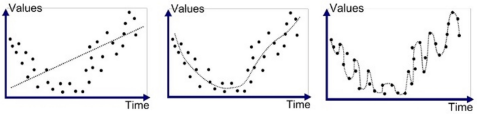
1. Idea: Given training data (x_i, y_i), find a $f: \mathcal{X} \to \mathcal{Y}$ such that $f(x_i) \approx y_i$, where
   - ① $x_i \in \mathcal{X} \subseteq \mathbb{R}^d$: the feature vector for the i^{th} training example
   - ② $y_i \in \mathcal{Y} \subseteq \mathbb{R}^t$: t responses
     - note we could have $t=1$ or even $t=\infty$
2. Note for any finite training data (x_i, y_i), $i=1,\dots,n$, there exist infinitely many functions f such that for all i, $f(x_i) = y_i$.

eg



3. Moreover, our prediction $\hat{y} = f(x)$ can vary significantly on new data x!
4. To choose f, we can
   - ① leverage prior knowledge of f; &
     - eg if x & y come from a population which follows "rules"
   - ② choose the "simplest" function.

UNDERFITTING, GOOD FITTING, OVERFITTING



Underfitted | Good Fit/Robust | Overfitted

STATISTICAL LEARNING

We assume the training & test data are both iid samples from the same unknown distribution $\mathcal{P}$; ie

$$(X_i, Y_i) \sim \mathcal{P}$$
$$(X, Y) \sim \mathcal{P}.$$

LEAST SQUARES REGRESSION

We want to choose f so that

$$f = \min_{f: \mathcal{X} \to \mathcal{Y}} \mathbb{E} \|f(X) - Y\|_2^2.$$

this is our least squared error.

REGRESSION FUNCTION: $m(x)$

1. Our "regression function" is

$$f^*(x) = m(x) = \mathbb{E}[Y \mid X = x].$$

2. However, calculating m requires us to know the distribution of P, ie all pairs (X, Y).

3. We show that m is optimal; ie

$$m(x) = \min_{f: X \to Y} E\|f(X) - Y\|_2^2.$$

Proof. First, see that

$$\begin{aligned} E\|f(X) - Y\|_2^2 &= E\|f(X) - m(X) + m(X) - Y\|_2^2 \\ &= E\|f(X) - m(X)\|_2^2 + E\|m(X) - Y\|_2^2 \\ &\quad + 2E\langle f(X) - m(X),\ m(X) - Y\rangle. \end{aligned}$$

(using $\|a+b\|_2^2 = \|a\|_2^2 + \|b\|_2^2 + 2\langle a, b\rangle$)

Then

$$\begin{aligned} &E_{X,Y}\left[\langle f(X) - m(X),\ m(X) - Y\rangle\right] \\ &= E_X\left[E_{Y|X}\left[\langle f(X) - m(X),\ m(X) - Y\rangle\right]\right] \\ &\quad \text{(by double expectation theorem, see STAT 330)} \\ &= E_X\left[\langle f(X) - m(X),\ m(X) - \underbrace{E[Y \mid X]}_{m(X)}\rangle\right] \\ &= E_X[\langle f(X) - m(X),\ 0\rangle] \\ &= 0. \end{aligned}$$

Hence

$$E\|f(X) - Y\|_2^2 = E\|f(X) - m(X)\|_2^2 + \underbrace{E\|m(X) - Y\|_2^2}_{\text{noise (variance) term — independent wrt } f}.$$

Therefore, to reduce $E\|f(X) - Y\|_2^2$, we need to only minimize $E\|f(X) - m(X)\|_2^2$, which is minimal (ie $= 0$) when $f = m$!

4. However, m is unaccessible since the conditional distribution is unknown, so we need to try to get close to m using the training data.

BIAS-VARIANCE TRADEOFF

1. Let f_D be the regressor learned on the training dataset D. Then

$$\underbrace{E_{D,X,Y}\|f_D(X) - Y\|_2^2}_{\text{test error}} = \underbrace{E_X\|E_D[f_D(X)] - m(X)\|_2^2}_{\text{bias}^2} + \underbrace{E_{D,X}\|f_D(X) - E_D[f_D(X)]\|_2^2}_{\text{variance}} + \underbrace{E_{X,Y}\|m(X) - Y\|_2^2}_{\text{noise (variance)}}$$

Proof. We have shown

$$E_{X,Y}\|f_D(X) - Y\|_2^2 = E_X\|f_D(X) - m(X)\|_2^2 + \underbrace{E_{X,Y}\|m(X) - Y\|_2^2}_{\text{noise — independent wrt } f_D}.$$

Taking E_D of both sides:

$$E_D E_{X,Y}\|f_D(X) - Y\|_2^2 = E_D E_X\|f_D(X) - m(X)\|_2^2 + E_{X,Y}\|m(X) - Y\|_2^2. \quad -①$$

Define $\bar{f}(x) = E_D[f_D(X)]$.

Idea: we can sample multiple f's from various samples D:

$$\left.\begin{aligned} D_1 \sim P &\to f_{D_1} \\ &\vdots \\ D_n \sim P &\to f_{D_n} \end{aligned}\right\} \text{ then we define } \bar{f}(x) = \text{avg } f_{D_i}(x).$$

Then

$$\begin{aligned} &E_D E_X\|f_D(X) - m(X)\|_2^2 \\ &= E_{D,X}\|f_D(X) - \bar{f}(X) + \bar{f}(X) - m(X)\|_2^2 \\ &= E_{D,X}\|f_D(X) - \bar{f}(X)\|_2^2 + E_{D,X}\|\bar{f}(X) - m(X)\|_2^2 \\ &\quad + 2E_{D,X}\langle f_D(X) - \bar{f}(X),\ \bar{f}(X) - m(X)\rangle. \end{aligned}$$

Similarly, see that

$$\begin{aligned} &E_{D,X}\langle \bar{f}(X) - f_D(X),\ m(X) - \bar{f}(X)\rangle \\ &= E_X E_D\langle \underbrace{m(X) - \bar{f}(X)}_{\text{constant wrt } D},\ \bar{f}(X) - f_D(X)\rangle \\ &= E_X\langle m(X) - \bar{f}(X),\ \bar{f}(X) - \underbrace{E_D[f_D(X)]}_{\bar{f}(X)}\rangle \\ &= 0. \end{aligned}$$

Expanding ① yields the result desired. ∎

2. In particular, as the model capacity increases,
   ① the bias term decreases (ie model is more expressively powerful); but
   ② the variance increases (ie model is less stable).

SAMPLING → TRAINING

1. In practice, we can only calculate the sample average, ie we find f so that

$$f = \min_{f:\mathcal{X}\to\mathcal{Y}} \hat{E}\,\|f(X)-Y\|_2^2 := \frac{1}{n}\sum_{i=1}^{n}\|f(X_i)-Y_i\|_2^2.$$

2. However, as our training data size $n\to\infty$, $\hat{E}\to E$ & hopefully $\operatorname{argmin}\hat{E} \to \operatorname{argmin} E$.

LINEAR REGRESSION

1. In linear regression, our regression functions are "affine"; ie in the form

$$f(x) = Wx + b, \quad W\in\mathbb{R}^{t\times d}, \quad b\in\mathbb{R}^{t}.$$

- t = # of response parameters we want to predict
- d = # of input parameters

2. Again, we can use padding:

$$x \leftarrow \begin{pmatrix} x \\ 1 \end{pmatrix}, \quad W \leftarrow [W, b] \Rightarrow f(x) = Wx$$

3. In matrix form:

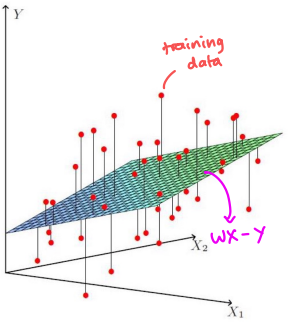
$$\frac{1}{n}\sum_i \|f(x_i)-y_i\|_2^2 = \frac{1}{n}\|WX-Y\|_F^2,$$

$$X\in[x_1,\dots,x_n]\in\mathbb{R}^{(d+1)\times n}, \quad Y=[y_1,\dots,y_n]\in\mathbb{R}^{t\times n},$$

$$\|A\|_F = \sqrt{\sum_{i,j} a_{ij}^2}$$

4. We want to find W such that

$$W = \min_{W\in\mathbb{R}^{t\times(d+1)}} \frac{1}{n}\|WX-Y\|_F^2.$$



- geometrically, we want to minimise the sum of distances between the input training data & the resultant hyperplane.

SOLVING LINEAR REGRESSION

1. We define our loss function as

$$\text{Loss}(W) = \frac{1}{n}\|WX-Y\|_F^2$$

2. Taking the derivative wrt W & setting to zero:

$$\nabla_W \text{Loss}(W) = \frac{2}{n}(WX-Y)X^T \;(=0)$$
$$\Rightarrow WXX^T = YX^T$$
$$\Rightarrow W = YX^T(XX^T)^{-1}$$

PREDICTION

1. Once we have solved W on the training set (X,Y), we can predict on unseen data X_{test}:

$$\hat{Y}_{test} = WX_{test}$$

2. The "test error" (if true labels were available) is

$$\text{test error} = \frac{1}{n_{test}}\|Y_{test} - \hat{Y}_{test}\|_F^2$$

3. The "training error" is

$$\text{training error} = \frac{1}{n}\|Y-WX\|_F^2.$$

4. We can minimize the training error to reduce the test error.

ILL-CONDITIONING

1. Consider $X=\begin{bmatrix}0 & \varepsilon \\ 1 & 1\end{bmatrix}$, $y=(1\ -1)$. Solving linear least squares regression:

$$W = YX^T(XX^T)^{-1} = (1\ -1)\begin{pmatrix}-1/\varepsilon & 1 \\ 1/\varepsilon & 0\end{pmatrix}$$
$$= (-2/\varepsilon\ \ 1)$$

2. So slight perturbation leads to chaotic behavior!

3. This occurs when X is ill-conditioned: ie close to rank deficient

ie
- two cols in X are close to linearly dependent
- but corresponding y's are different
- this is a contradiction ⇒ W becomes unstable.

RIDGE REGRESSION

💡1 **Idea**: We instead try to find

$$W = \min_W \left[\frac{1}{n} \|WX - Y\|_F^2 + \lambda \|W\|_F^2 \right]$$

Why is this better?

Consider $\text{Loss}(W) = \frac{1}{n} \|WX - Y\|_F^2 + \lambda \|W\|_F^2$.

$$\Rightarrow \nabla_W \text{Loss}(W) = \frac{2}{n}(WX - Y)X^T + 2\lambda W \quad (=0)$$

$$\Rightarrow WXX^T - YX^T + \lambda n W = 0$$

$$WXX^T - YX^T + W(\lambda n I) = 0$$

$$WXX^T + W(\lambda n I) = YX^T$$

$$\therefore W = (XX^T + \lambda n I)^{-1}(YX^T)$$

Then $XX^T + n\lambda I$ is far from rank-deficient matrices for large λ. (Proof uses SVD - see MATH 235).

💡2 λ controls our **trade-off**:

1. $\lambda = 0$ reduces to **ordinary linear regression**;
2. $\lambda = \infty$ reduces to $W \equiv 0$; &
3. intermediate λ restricts output to be $\frac{1}{\lambda}$ proportional to input.

💡3 Alternatively, note

$$\frac{1}{n} \|WX - Y\|_F^2 + \lambda \|W\|_F^2 = \frac{1}{n} \| W[X \ \sqrt{n\lambda I}] - [Y \ 0] \|_F^2$$

So we can also

1. augment X with $\sqrt{n\lambda I}$; ie $\widetilde{X} = (X \ \sqrt{n\lambda I})$
2. augment Y with zeroes; ie $\widetilde{Y} = (Y \ 0)$

(ie **data augmentation**) to achieve **regularization**.

Chapter 3: Logistic Regression

MOTIVATION

1. This is for **linear classification**.
2. We can use $|\langle x, w \rangle|$ (our **margin**) as a measure of our confidence in the prediction $\hat{y}$.
3. However, as this is **un-normalized**, it is hard to interpret.

MLE

1. We want to directly learn our "confidence"

$$p(x; w) := P(Y=1 \mid X=x)$$

2. Then, if $Y_1, \dots, Y_n, X_1, \dots, X_n$ are independent, then

$$\begin{aligned} & P(Y_1=y_1, \dots, Y_n=y_n \mid X_1=x_1, \dots, X_n=x_n) \\ &= \prod_{i=1}^{n} P(Y_i=y_i \mid X_i=x_i) \\ &= \prod_{i=1}^{n} [p(x_i; w)]^{y_i} [1-p(x_i; w)]^{1-y_i} \quad \text{if } Y=\{0,1\} \end{aligned}$$

3. Maximizing the likelihood:

$$\begin{aligned} & \max_w \prod_{i=1}^{n} [p(x_i; w)]^{y_i} [1-p(x_i; w)]^{1-y_i} \\ \Leftrightarrow & \min_w \sum_{i=1}^{n} [-y_i \log p(x_i; w) - (1-y_i) \log(1-p(x_i; w))] \end{aligned}$$

4. We thus want to find w which satisfies the above optimization problem.

THE LOGIT TRANSFORM

1. If we assume the log of odds ratio is linear: ie

$$\log \frac{p(x; w)}{1-p(x; w)} = \langle x, w \rangle$$

* we can only perform logistic regression if we assume this!

then

$$p(x; w) = \frac{1}{1+\exp(-\langle x, w \rangle)}$$

↳ this is also called the "sigmoid transformation".

2. Plugging this into the earlier optimization problem, we want to find

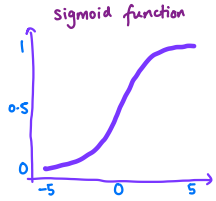
$$\min_w \sum_{i=1}^{n} \log[1+\exp(-\langle x_i, w \rangle)] + (1-y_i)\langle x_i, w \rangle$$

if $y_i \in \{0,1\}$.

3. If instead $y_i \in \{\pm 1\}$, then

$$\min_w \sum_{i=1}^{n} \log[1+\exp(-y_i \langle x_i, w \rangle)]$$

↳ this is "logistic loss".



TRAINING LOGISTIC REGRESSION

Our gradient descent algorithm is

$$w \leftarrow w - \eta \nabla_w \text{Loss}(w)$$

PREDICTION

We take

$$\hat{y} = 1 \iff P(Y=1 \mid X=x) > \frac{1}{2} \iff \langle x, w \rangle > 0$$

Our decision boundary is still

$$H := \{ x : \langle x, w \rangle = 0 \}$$

So we can predict $\hat{y} = \text{sign}(\langle x, w \rangle)$ as before, but now with confidence $p(x; w)$.

MULTI-CLASS EXTENSION

Idea: For a class $y \in \{1, \dots, c\}$, we want to learn $\{w_1, \dots, w_c\}$ for each class.

We consider the "softmax" function:

$$P(Y=k \mid X=x, W=[w_1, \dots, w_c]) = \frac{\exp(\langle x, w_k \rangle)}{\sum_{\ell=1}^{c} \exp(\langle x, w_\ell \rangle)}$$

- non-negative and sum to 1

Logits	$\langle x, w_1 \rangle$	$\langle x, w_2 \rangle$	... ...	$\langle x, w_c \rangle$

Softmax operation

Probability (confidence)	$\frac{\exp(\langle x, w_1 \rangle)}{\sum_i \exp(\langle x, w_i \rangle)}$	$\frac{\exp(\langle x, w_2 \rangle)}{\sum_i \exp(\langle x, w_i \rangle)}$	... ...	$\frac{\exp(\langle x, w_c \rangle)}{\sum_i \exp(\langle x, w_i \rangle)}$

- we map a real-valued vector to a probability vector
- these are non-negative & sum to 1.

Training: again, we use MLE:

$$\min_w E\left[-\log \frac{\exp(\langle X, w_Y \rangle)}{\sum_{\ell=1}^{c} \exp(\langle X, w_\ell \rangle)} \right]$$

Prediction:

$$\hat{y} = \underset{k}{\text{argmax}}\ P(Y=k \mid X=x;\ W=[w_1, \dots, w_c]$$

Chapter 4: Hard-Margin Support Vector Machines

INTRODUCTION

1. We assume $y = \{-1, +1\}$, and don't use padding.
2. Perceptron: we find any $w \in \mathbb{R}^d$, $b \in \mathbb{R}$ such that

$$\min_{w,b} 0 \quad \text{s.t.} \quad y_i \hat{y}_i > 0 \ \forall i, \qquad \hat{y}_i = \langle x_i, w \rangle + b$$

$$\Leftrightarrow \min_{w,b} 0 \quad \text{s.t.} \quad y_i \hat{y}_i \geq 1 \ \forall i$$

3. However, the larger the margin, the faster Perceptron converges.

recall # mistakes, $M \leq \frac{C^2}{\gamma^2}$, $\|x_i\|_2 \leq C$, $\gamma = \min_i |\langle x_i, w^* \rangle|$, $\|w^*\|_2 = 1$.

4. So, the goal of hard-margin SVM is to maximize the margin (assuming data is linearly separable).

DISTANCE FROM A POINT TO A HYPERPLANE

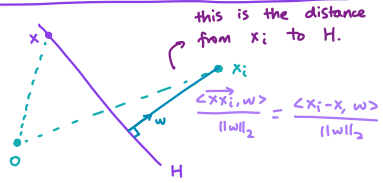
Let $H := \{x : \langle x, w \rangle + b = 0\}$. Then

$$\text{distance}(x_i, H) = \frac{|\langle x_i - x, w \rangle|}{\|w\|_2}, \quad x \in H$$

$$= \frac{|\langle x_i, w \rangle - \langle x, w \rangle|}{\|w\|_2}$$

$$= \frac{|\langle x_i, w \rangle + b|}{\|w\|_2} \quad \because x \in H$$

$$= \frac{y_i \hat{y}_i}{\|w\|_2} \quad \because y_i \hat{y}_i > 0$$

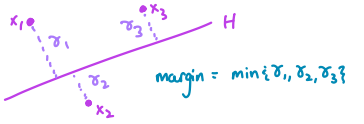


MARGIN

1. We define the "margin" as the smallest distance to a separating hyperplane H among all separable training data; ie

$$\text{margin} = \min_i \frac{y_i \hat{y}_i}{\|w\|_2} = \min_i \frac{|\langle x_i, w \rangle + b|}{\|w\|_2}, \qquad H = \{x : \langle x, w \rangle + b = 0\}$$

eg



2. Our goal is to maximize the margin among all hyperplanes: ie find

$$\max_{w,b} \min_i \frac{y_i \hat{y}_i}{\|w\|_2} \quad \text{s.t.} \quad y_i \hat{y}_i > 0 \ \forall i$$

TRANSFORMING TO STANDARD FORM

$_1$ Note for the margin, (w,b) & (cw,cb) has the same loss for $c>0$.

$_2$ So, we can fix the numerator arbitrarily to 1:

$$\max_{w,b}\left[\frac{1}{\|w\|_2} \quad \text{s.t.} \quad \min_i y_i \hat{y}_i = 1\right]$$

$$\Rightarrow \min_{w,b}\left[\frac{1}{2}\|w\|_2^2 \quad \text{s.t.} \quad y_i(\langle x_i, w\rangle + b) \geq 1 \quad \forall i\right]$$

COMPARISON TO PERCEPTRON

$_1$ **Hard-margin SVM**

$$\min_{w,b} \frac{1}{2}\|w\|_2^2 \quad \text{s.t.} \quad y_i\hat{y}_i \geq 1 \quad \forall i$$

- quadratic programming
- unique solution
- maximal margin

Perceptron

$$\min_{w,b} 0 \quad \text{s.t.} \quad y_i\hat{y}_i \geq 1 \quad \forall i$$

- linear programming
- infinitely many solutions
- convergence rate depends on max margin

SUPPORT VECTORS

$_1$ Note that

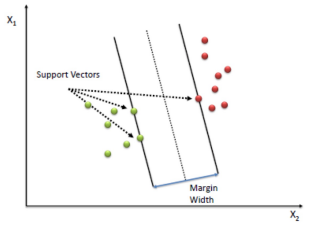
$$y_i\hat{y}_i \geq 1 \quad \forall i \iff \begin{cases} \hat{y}_i \geq +1 & \forall i: y_i = +1 \\ \hat{y}_i \leq -1 & \forall i: y_i = -1 \end{cases}$$

$_2$ This yields 3 parallel hyperplanes:

$$H = \{x: \langle x, w\rangle + b = 0\}$$
$$H^+ = \{x: \langle x, w\rangle + b = +1\}$$
$$H^- = \{x: \langle x, w\rangle + b = -1\}$$

} the "supporting" hyperplanes

$_3$ "Support vectors" are those where points lie on the supporting hyperplanes.



LAGRANGIAN DUAL

First, we show

$$\min_{w,b} \frac{1}{2}\|w\|_2^2 \quad \text{s.t.} \quad y_i(\langle x_i, w\rangle + b) \geq 1 \quad \forall i$$

$$= \min_{w,b} \max_{\alpha \geq 0} \frac{1}{2}\|w\|_2^2 - \sum_i \alpha_i [y_i(\langle x_i, w\rangle + b) - 1]$$

$\alpha = [\alpha_1, \dots, \alpha_n] \in \mathbb{R}^n$;
$\alpha \geq 0 \iff \alpha_i \geq 0 \ \forall i$

Proof. Let Δ be the second expression.

See that

$$\Delta = \min_{w,b} \max_{\alpha \geq 0} \frac{1}{2}\|w\|_2^2 - \sum_i \alpha_i [y_i(\langle x_i, w\rangle + b) - 1]$$

If $\exists i$ s.t. $y_i(\langle x_i, w\rangle + b) < 1$, then if we set $\alpha_i = \infty$, it follows that $\Delta = +\infty$, which is the maximal value Δ can take.

Otherwise, ie if $\forall i$, $y_i(\langle x_i, w\rangle + b) \geq 1$, then

$$\Delta = \frac{1}{2}\|w\|_2^2 - \sum_i \underbrace{\alpha_i}_{\text{+ve}} \underbrace{[y_i(\langle x_i, w\rangle + b) - 1]}_{\text{+ve}}$$

$$\leq \frac{1}{2}\|w\|_2^2.$$

If we set $\alpha_i = 0 \ \forall i$, we get $\Delta = \frac{1}{2}\|w\|_2^2$, which is the max value Δ can take.

Therefore,

$$\Delta = \min_{w,b} \begin{cases} +\infty, & \text{if } \exists i \text{ s.t. } y_i(\langle x_i, w\rangle + b) < 1 \\ \frac{1}{2}\|w\|_2^2, & \text{otherwise} \end{cases}$$

$$= \min_{w,b} \frac{1}{2}\|w\|_2^2 \quad \text{if} \quad y_i(\langle x_i, w\rangle + b) \geq 1$$

as needed. ∎

We can swap the min & max:

$$\max_{\alpha \geq 0} \min_{w,b} \frac{1}{2}\|w\|_2^2 - \sum_i \alpha_i [y_i(\langle x_i, w\rangle + b) - 1]$$

(because of "strong duality")

Now, suppose we fix α, and consider the inner minimization problem.

Then w, b minimizes the function if $\frac{\partial}{\partial w} = \frac{\partial}{\partial b} = 0$.

Let $\text{Loss}(w, b) = \frac{1}{2}\|w\|_2^2 - \sum_i \alpha_i [y_i(\langle x_i, w\rangle + b) - 1]$.

$$\Rightarrow \frac{\partial}{\partial w} = w - \sum_i \alpha_i y_i x_i \ (=0), \quad \frac{\partial}{\partial b} = -\sum_i \alpha_i y_i \ (=0)$$

$$\rightarrow w = \sum_i \alpha_i y_i x_i, \quad \sum_i \alpha_i y_i = 0.$$

Finally, we consider the "outer" maximization problem.

Plugging in our value of w above:

$$\Rightarrow \text{Loss}(\alpha) = \frac{1}{2}\left\|\sum_i \alpha_i y_i x_i\right\|_2^2 - \left\langle \sum_i \alpha_i y_i x_i, \sum_i \alpha_i y_i x_i \right\rangle - b \underbrace{\sum_i \alpha_i y_i}_{0} + \sum_i \alpha_i$$

$$= -\frac{1}{2}\left\|\sum_i \alpha_i y_i x_i\right\|_2^2 + \sum_i \alpha_i \quad \text{s.t.} \quad \sum_i \alpha_i y_i = 0$$

Thus, our problem becomes

$$\max_{\alpha \geq 0} \sum_i \alpha_i - \frac{1}{2}\left\|\sum_i \alpha_i y_i x_i\right\|_2^2 \quad \text{s.t.} \quad \sum_i \alpha_i y_i = 0$$

$$= \min_{\alpha \geq 0} -\sum_i \alpha_i + \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \quad \text{s.t.} \quad \sum_i \alpha_i y_i = 0$$

WHY USE THE DUAL FORM?

Idea: If data is not linearly separable, we use a non-linear mapping ϕ to map the data.

$$\min_{\alpha \geq 0} -\sum_i \alpha_i + \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle \phi(x_i), \phi(x_j) \rangle$$

$$\text{s.t.} \quad \sum_i \alpha_i y_i = 0.$$