$(f(x_i))$ decreasing $\Rightarrow f(x_k) - f(x^*) \leq f(x_i) - f(x^*) \quad \forall i \leq k$

$$\therefore k(f(x_k) - f(x^*)) = \sum_{i=1}^{k} (f(x_k) - f(x^*))$$
$$\leq \sum_{i=1}^{k} (f(x_i) - f(x^*))$$
$$\leq \frac{1}{2\alpha} \|x_0 - x^*\|_2^2$$

∎

* Lecture 9

Sec. 4.4 Constrained optimization

Consider $f: \mathbb{R}^n \to \mathbb{R}$ and $S \subseteq \mathbb{R}^n$ and find

$$x^* = \operatorname{argmin}_{\vec{x} \in S} f(\vec{x}) \quad \text{or} \quad f_{min} = \min_{\vec{x} \in S} f(\vec{x}) \quad \left(\text{or } \max\right)$$

(same problem)

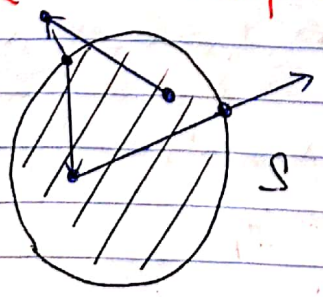
E.g. $A \in \mathbb{R}^{m \times n}$ and consider

$$f(\vec{x}) = \|A\vec{x}\|_2 \qquad f: \mathbb{R}^n \to \mathbb{R}$$

$$\max_{\vec{x}: \|\vec{x}\|_2 = 1} f(\vec{x}) = \max_{\vec{x}: \|\vec{x}\|_2 = 1} \|A\vec{x}\|_2$$
$$= \|A\|_2.$$

• one approach (modification of gradient descent)

Idea: At each step, "project" back into S.



• more general

The Kanush-Kuhn-Tucker approach (generalizes Lagrange multipliers)

negation: $\vec{x} \notin S \Rightarrow g_i(\vec{x}) \neq 0$ some i or $h_j(\vec{x}) > 0$ some j

(*) If we can write $S = \{\vec{x} \mid g_i(\vec{x}) = 0$ and $h_j(\vec{x}) \leq 0$ $1 \leq i \leq n$, $1 \leq j \leq m\}$ (number of conditions you have)
for some functions $g_i, h_j : \mathbb{R}^n \to \mathbb{R}$.

Define the Lagrangian

$L(\vec{x}; \vec{\lambda}, \vec{\alpha}) = f(\vec{x}) + \sum_i \lambda_i g_i(\vec{x}) + \sum_j \alpha_j h_j(\vec{x})$.

$\vec{\lambda}^T \vec{g}(\vec{x}) = \sum_i \lambda_i g_i(\vec{x})$

$\vec{\alpha}^T \vec{h}(\vec{x})$

Then $\min_{x \in \mathbb{R}^n} \max_{\vec{\lambda}} \max_{\vec{\alpha} : \vec{\alpha} \geq 0} L(\vec{x}; \vec{\lambda}, \vec{\alpha})$ (*)

has the same solutions $\vec{x}$ as $\min_{\vec{x} \in S} f(\vec{x})$.

NOTE USE

Justification: Anytime $\vec{x} \notin S \Rightarrow g_i(\vec{x}) \neq 0$

$\max_{\vec{\lambda}} \max_{\vec{\alpha} : \vec{\alpha} \geq 0} L(\vec{x}; \vec{\lambda}, \vec{\alpha}) = \infty$ unbounded. ∵ can make $\vec{\lambda}$ as large as we want.

and whenever $\vec{x} \in S$

$\max_{\vec{\lambda}} \max_{\vec{\alpha} : \vec{\alpha} \geq 0} L(\vec{x}; \vec{\lambda}, \vec{\alpha}) = f(\vec{x})$ ∵ $g_i(\vec{x}) = 0$, $h_j(\vec{x}) \leq 0$. ∴ vanishing best to pick $\alpha_j = 0$

Theorem [Karush, Kuhn, Tucker]: Let f be differentiable. (1939 — 1951)

The following conditions are necessary for $\vec{x}$ to be a solution to (*)

① $\nabla_{\vec{x}} L(\vec{x}; \vec{\lambda}, \vec{\alpha}) = 0$

② ~~$\vec{x} \in S$~~

③ $\sum_i \alpha_i h_i(\vec{x}) = 0$.

gradients could be a set of possible slopes. (i.e. set of subderivatives)

E.g. Let $A \in \mathbb{R}^{n \times n}$ be symmetric.

$f(\vec{x}) = -\vec{x}^T A \vec{x}$ find $\min_{\vec{x} : \|\vec{x}\|_2 \leq 1} f(\vec{x})$

$h(\vec{x}) = \|x\|_2^2 - 1 = \vec{x}^T \vec{x} - 1 \leq 0$ ← How to get?

$L(\vec{x}; \alpha) = -\vec{x}^T A \vec{x} + \alpha(\vec{x}^T \vec{x} - 1)$

① $\Rightarrow \nabla_{\vec{x}} L(\vec{x}; \alpha) = 0$

$\Rightarrow -\vec{x}^T (A + A^T) + 2\alpha \vec{x}^T = 0$

$\Rightarrow -2\vec{x}^T A = 2\alpha \vec{x}^T$ taking transpose on both sides.

$\Rightarrow A\vec{x} = \alpha \vec{x}$ ↓ so $\vec{x}$ is an eigenvector of A with eigenvalue α

$\Uparrow$ ∵ $A^T = A$. $(\alpha A)^T = A^T \alpha^T = \alpha A^T$. α is a constant / 1×1 matrix

usually the most significant one

$A = B \Rightarrow A^T = B^T$

e.g. If $AB = CDE$

$\Rightarrow B^T A^T = E^T D^T C^T$

* $\vec{x}^T\vec{y} = \vec{y}^T\vec{x}$

$\sum_i x_i y_i \quad \sum_i y_i x_i$

$\vec{x} \cdot \vec{y}$

② $\|x\|_0 \leq 1 \quad \vec{x} \in S = \{\vec{x} \mid \|\vec{x}\|_2 - 1 = 0\} \Rightarrow \|\vec{x}\|_2 - 1 = 0 \Rightarrow \|\vec{x}\|_2 = 1$

③ $\|x\|_0 = 1 \quad \alpha h(\vec{x}) = 0 \Rightarrow \alpha(\|\vec{x}\| - 1) = 0$ ∴ either $\alpha = 0$ or $\|\vec{x}\| = 0$

solution: $\vec{x}$ is the (normalized) eigenvector of A with the largest eigenvalue. ($f(\vec{x}) = -\alpha$)

$f(\vec{x}) = -\alpha$ when $\vec{x}$ is an EV of EV α.

* Chapter 5 Basics of ML.

"Definition" A learning algorithm is a computer program which accomplishes some task T based on some experience E with respect to some performance measure P, such that performance P increases as we add experience to E

(not formal) understand what is meant by E, T, P by example.

- Experience E: Usually a dataset $X \in \mathbb{R}^{m \times n}$ where each column is called a feature and each row is called an example.

E.g. Suppose we have 150 iris plants.

	petal length	petal width	sepal length	sepal width
plant 1	2.5	1.2	4.6	2.6
plant 2	3	.5	3.4	4.3
plant 3	4.2	1.3	4.5	1.2
⋮	⋮			

very diff- (cm)

← petals

↑ sepals

about the ~~abotth~~ same.

3 different species known 1, 2, 3.

E may also consist of a vector y of labels.

E.g. $y = \begin{bmatrix} 1 \\ 3 \\ 2 \\ \vdots \end{bmatrix}$

A learning algorithm with X, y is called supervised.

" " " X " " unsupervised.

- Unsupervised Goal determine a probability distribution $p(x)$ where $x \in \mathbb{R}^n \leftrightarrow X_{i,*}$.

- supervised usually want to determine $p(y|x)$.

If y takes a discrete values (e.g. $\{1, \ldots, k\}$) → classification.

If y takes continuous " (e.g. $\mathbb{R}$ or $[a, b] \subseteq \mathbb{R}$) → regression

while logistic regression is a classification also.

☆Lecture 10

(E.T.P, program).

• Tasks T: ① classification Given (X, y) $X \in \mathbb{R}^{m \times n}$ $y \in \mathbb{R}^m$ $y_i \in \{1, \dots, k\}$.
(Examples) produce $f: \mathbb{R}^n \to \{1, \dots, k\}$ s.t. $f(x) = y$ performs as well as a prediction.

E.g. object recognition $x_{i,*}$ = pixel data of an image.
y_i = cat or dog

② Regression Given (X, y), $y_i \in \mathbb{R}$
task: produce $f: \mathbb{R}^n \to \mathbb{R}$ s.t. $f(x) = y$ performs well as a prediction

E.g. (Predicting Housing Prices)
$x_{i,*}$ = data about a house.
y_i = price of house

Each task would require a very different type of algorithm.

③ Given an image of text, produce text in Unicode (digital).

④ Translate from one language to another.

⑤ Generate new examples similar to those in E.

⑥ Given X with some entries missing, fill in entries (as reliably as possible).

⑦ Given X, remove noise (SVD often useful)

• Performance Measure P

Depends on T (the specific task you are doing).
① (classification) Accuracy / Precision / Recall
different types of performance measure for classification

② (Regression) Average (squared) error in prediction.

★ Performance is not measured using examples from E but from a different set of examples. $X^{(test)}$, $y^{(test)}$

• Program

Examples: ② (Regression) Linear Regression

Note: using same idea gives us polynomial regression.

E: (supervised) Dataset $X \in \mathbb{R}^{m \times n}$ labels $y \in \mathbb{R}^m$. $y_i \in \mathbb{R}$

T: Produce $f: \mathbb{R}^n \to \mathbb{R}$

P: Given test set $X^{(test)}$, $y^{(test)}$ m = # examples

~~want to minimize~~

case ①: Expect y_i depends linearly (affinely) on $x_{i,*}$

so $\exists w$ s.t. $y_i \approx x_{i,*} w + b$.

then, we want to minimize. Introduce $\hat{X} = \begin{bmatrix} 1 & \\ \vdots & X \end{bmatrix}$ so no need to have $+b$

$$\frac{1}{m} \sum_{i=1}^{m} \| \hat{X}^{(test)}_{i,*} w - y_i^{(test)} \|_2$$

$$= \frac{1}{m} \| \hat{X}^{(test)} w - y^{(test)} \|_2^2$$

should always be evaluated on the test sets.

Define the cost function (Loss, Error, ...)

$$J(w) = \frac{1}{m} \| \hat{X} w - y \|_2^2$$ ← (cost function defined using E.)

More justification in §5.5.1

Find $w^* = \arg\min_w J(w)$.

solution 1: Use gradient descent to approximate w^*.

~~proposition~~: J is twice differentiable, convex, and has Lipschitz continuous gradient. Furthermore

① $\nabla_w J(w) = \frac{2}{m} [\hat{X}^T \hat{X} w - \hat{X} y]$

② $H(J)(w) = \frac{2}{m} \hat{X}^T \hat{X}$

proof: HW

solution 2: proposition If $\hat{X}$ has LI columns and J has a global minimum, w^*, then $w^* = (\hat{X}^T\hat{X})^{-1}\hat{X}^T y$

proof: since J is differentiable, w^* global min $\Rightarrow$ w^* local min $\Rightarrow \nabla_w J(w^*) = 0$.

$$\nabla_w J(w^*) = \frac{2}{N}[\hat{X}^T\hat{X}w - \hat{X}^T y] = 0$$

$$\Rightarrow \hat{X}^T\hat{X}w = \hat{X}^T y$$

(since columns of $\hat{X}$ are LI, $\hat{X}^T\hat{X}$ is pos def $\Rightarrow \hat{X}^T\hat{X}$ is invertible.)

$$\Rightarrow w = (\hat{X}^T\hat{X})^{-1}\hat{X}^T y$$

hard to invert a matrix if the matrix is large

If $\hat{X}^T\hat{X}$ not invertible, then $\exists w \neq 0$ s.t. $\hat{X}^T\hat{X}w = 0$

by rank nullity.

then $w^T\hat{X}^T\hat{X}w = 0$ ✗ pos def.

note: use

case ② we expect y_i is a polynomial function $x_{i,*}$ of maximum total degree n.

so $f(x_1, \ldots, x_m) = f(\vec{x}) = \sum_{\alpha \in \mathbb{Z}^n_{\geq 0}} a_\alpha x^\alpha$ E.g. $\alpha = (0,1,2,3)$ $x^\alpha = x_1 x_3^2 x_4^3$

solution: modify X by adding new columns, one for each $\alpha \in \mathbb{Z}^n_{\geq 0}$ s.t. $1 \leq |\alpha| \leq n$ $\quad |\alpha| = \sum_i \alpha_i$ (total degree)

E.g. $X = \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 3 & 4 \end{bmatrix}$ $\xrightarrow[\text{deg } 2]{\max \text{ total}}$ $\begin{bmatrix} x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 \\ 1 & 2 & 1 & 2 & 4 \\ 2 & 1 & 4 & 2 & 1 \\ 3 & 4 & 9 & 12 & 16 \end{bmatrix}$

again, form $\hat{X}$ and proceed as in Case 1.

find coefficient w that minimize prediction error.

扫描全能王 创建

*Lecture 11

Example ① Logistic Regression

E: $X \in \mathbb{R}^{m \times n}$, $y \in \{0,1\}^m$ (supervised)

I: Produce a function $f: \mathbb{R}^n \to [0,1]$

from this, we define $\text{pred}_f: \mathbb{R}^n \to \{0,1\}$.

$$\text{pred}_f(\vec{x}) = \begin{cases} 1 & f(\vec{x}) > .5 \\ 0 & f(\vec{x}) < .5 \end{cases}$$

P: Many choices exist. Lets go with accuracy.

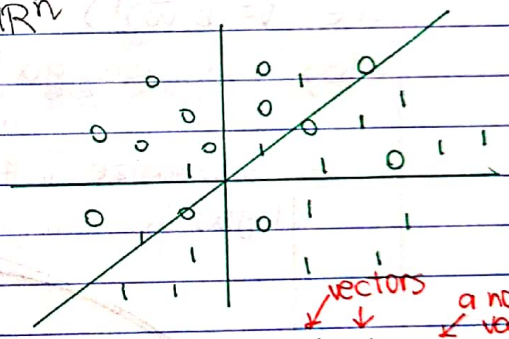
$$\text{Acc} = \frac{\#\{i \mid \text{pred}_f(X_{i,*}^{(\text{test})}) = y_i^{(\text{test})}\}}{\#\text{ rows in } X^{(\text{test})}}$$

Note: Also could use precision or recall

A solution: case ① we expect $y=1$ examples are reliably separated from $y=0$ examples by a hyperplane in $\mathbb{R}^n$

E.g. $X \in \mathbb{R}^{m \times 2}$

$$X = \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 4 & 3 \\ \vdots & \vdots \end{bmatrix} \quad y = \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix}$$



Goal: Find $\vec{w} \in \mathbb{R}^n$ $b \in \mathbb{R}$ s.t. the hyperplane $\vec{w}^T\vec{x} + b = 0$ separates 1's from 0's reliably.

vectors, a normal vector, equation for a hyperplane

we need the b, so the equation for the hyperplane does not go through the origin.

Question: How to find the best $\vec{w}, b$ using an optimization algorithm?

Want function $J(\vec{w}, b)$ which is minimized when accuracy is maximized. (cost function)

$\vec{x} \cdot \vec{w} + b \gg 0 \implies$ strong predict for 1.

$\vec{x} \cdot \vec{w} + b \ll 0 \implies$ strong predict for 0.

we want these.

This suggests we use a function g with the following properties

$$\lim_{z \to \infty} g(z) = 1.$$

$$g(0) = 1/2.$$

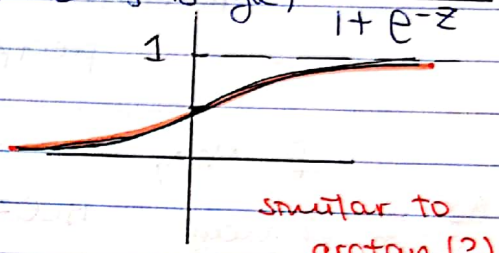
$$\lim_{z \to -\infty} g(z) = 0.$$

with this, we can compute $g(\vec{x}\cdot\vec{w}+b)$ and predict 1 if >0.5, 0 if <0.5

Defn: The sigmoid function $g:\mathbb{R}\to[0,1]$ is $g(z)=\frac{1}{1+e^{-z}}$

prop: ① g satisfies (+)

② $g'(z)=g(z)(1-g(z))$



proof: Exercise.

Again, we use $\hat{X}=\begin{bmatrix}1 & \\ \vdots & X \\ 1 & \end{bmatrix}$ so $\hat{X}\cdot\vec{v}$ instead of $XW+\begin{bmatrix}b\\ \vdots \\ b\end{bmatrix}$

(i.e. $V=(b\ \vec{w})^T$)

Idea: encourage $g(\vec{x}^T\vec{v})$ to be far from 0 if $y=1$.

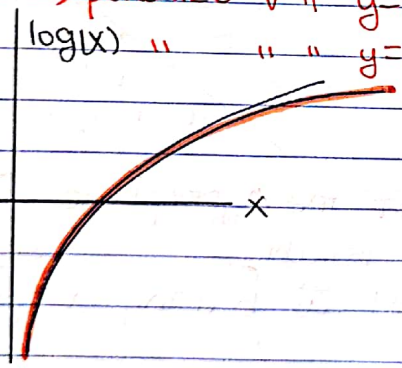
" " " " 1 " $y=0$.

$\Longleftrightarrow$ penalize $\vec{v}$ if $y=1$ and $-\log(g(\vec{x}^T\vec{v}))$ is large.

$\log(x)$ " " " $y=0$ " $-\log(1-g(\vec{x}^T\vec{v}))$ is large

$\log(x)=\ln(x)$

log = ln



combine the two cases into one big function

$\Longleftrightarrow$ penalize v if $-(1-y)\log(1-g(\vec{x}^T\vec{v}))-y\log(g(\vec{x}^T\vec{v}))$ is large

solution: define the cost function

[Given dataset $X,y = X^{(train)}, y^{(train)}$ * $X^{(test)}, y^{(test)}$ the test data]

$$J(v)=\frac{1}{m}\sum_i -(1-y_i)\log(1-g(\hat{X}_{i,*}^T\vec{v}))-y_i\log(g(\hat{X}_{i,*}^T\vec{v}))$$

$$=\frac{1}{m}\left[-(\vec{1}-\vec{y})^T\mathrm{Log}(\vec{1}-G(\hat{X}\vec{v}))-\vec{y}^T\mathrm{Log}(G(\hat{X}\vec{v}))\right]$$

where $\mathrm{Log}:\mathbb{R}^m\to\mathbb{R}^m$ $\mathrm{Log}(x_1,\dots,x_m)=(\log(x_1),\dots,\log(x_m))$

$G: \mathbb{R}^m \to \mathbb{R}^m \quad G(x_1, \dots, x_m) = (g(x_1), \dots, g(x_m))$

and find $v^* = \arg\min_v J(v)$ using some form of gradient descent

proposition: ① $\nabla_v J(v) = \frac{1}{m}(G(\hat{X}\vec{v}) - \vec{y})^T \hat{X}$

$$H(J)(v) = \frac{1}{m}\hat{X}^T D(v) \hat{X}$$

$$D(v) = \text{diag}(G'(\hat{X}v))$$

$$G'(x_1, \dots, x_m) = (g'(x_1), \dots, g'(x_m))$$

② J is twice differentiable convex, and has Lipschitz continuous gradient.

proof: HW2. ∎

∴ Gradient descent reliably finds the global min (if it exists)

case②: Expect a nonlinear "decision boundry"

solution: modify X by adding new columns corresponding to polynomial or other functions of current columns.

4.5.2 Capacity, Overfitting, Underfitting

Recall: Learning algorithms (E, T, P, program)

the program is "trained" using E (e.g. $x^{(train)}, y^{(train)}$)

but P is determined by "test" data (not in E).

(e.g. $x^{(test)}, y^{(test)}$) for supervised algo.

Usually have some "cost function" and we minimize this w.r.t training set to determine program.

That is, minimize "test error" by minimizing "training error".

Question: How can we statistically gaurantee this approach works?

Need to make assumptions (i.i.d assumptions): independent identically distributed

Both the training set and test set are gotten by independently randomly sampling the same probability distribution. (call this Pdata).

Usually our task T is accomplished by choosing some set $\vec{w}$ of parameters found by minimizing some cost function $J(\vec{w})$ on training set.

E.g.