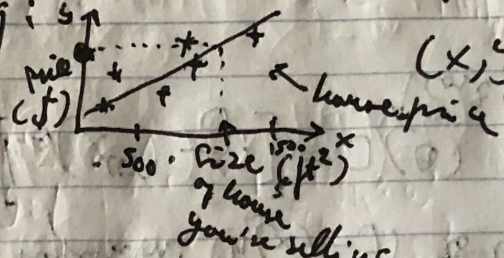
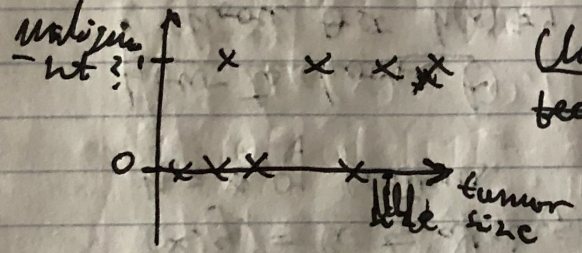
CS229 Lecture 1

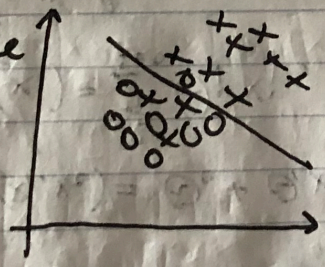
- First useful ML result was A. Samuel's 1959 Checkers AI.
- Well-posed learning problem: for problem A, computer learns from exp E wrt task T acc. to perf. measure P, if its perf. on T, as measured by P, improves with E.
- Supervised learning: (x, y): Learn $x \to y$



Regression problem because y is continuous.



Classification problem because y is discrete

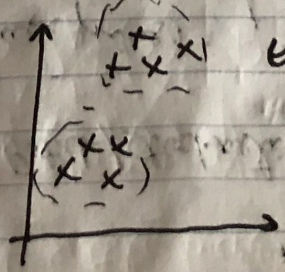


Predict whether tumor malign or benign
other features: clump thickness, uniformity of cell size, adhesion etc.

Support vector machine: inf num of features, ∞-dim vector to rep. a patient ... :O

Future subjects:
- ML Strategy (learning theory)
- Deep learning

Unsupervised learning: Basically, finds structure and patterns in data.



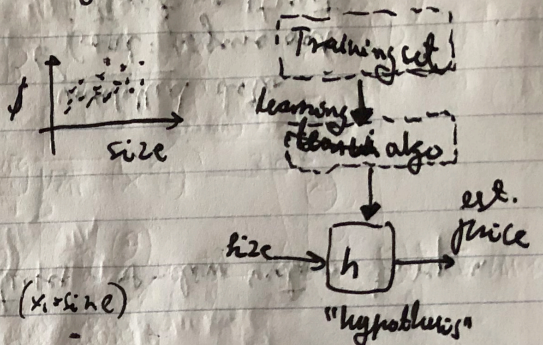
unlabelled data
Goal: figure out interesting structure here.

Reinforcement learning: Reward good behaviour, punish bad.

Lecture 2

Outline
• Linear regression; • Batch/stochastic gradient descent; • Normal Eq.

Take house prices:



How to represent h?
Let's say $h(x) = \theta_0 + \theta_1 x$

Can add bedrooms as a feature, x_2: (x_1, x_2)

Set notation: $\Rightarrow h(x_1, x_2) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$

"parameters" to be trained

$\Leftrightarrow h(x) = \sum_{j=0}^{2} \theta_j x_j$, $x_0 = 1 \Leftrightarrow \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}$, $x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$ $[h_\theta(x) = h(x)]$

def:
m = # training examples; x = "inputs/features"; y = "output/targ. variable"
(x, y) = a training example; $(x^{(i)}, y^{(i)})$ = ith training example
n = # of features

How to choose θ

• choose θ st $h(x) \approx y$ in training example
• (in ordinary linear regression: $(h_\theta(x) - y)^2$ to be minimised for θ.

$\Leftrightarrow \min_\theta \frac{1}{2} \sum_{i=1}^{m} [h_\theta(x^{(i)}) - y_i]^2 = J(\theta)$
↑ cost func / obj func

Gradient descent

• Start with some initial θ (say $\vec{\theta} = 0$)
• Keep changing θ to reduce $J(\theta)$
• Maths:

assignment

$\theta_j := \theta_j - \alpha \frac{\partial J}{\partial \theta_j}$; $a := a+1$; $a = b$ normal equality
(α = learning rate; := increment by one)

for $j = \{0, 1, 2\}$ (up to number of features)

for one training example. Can ignore the sum since derivation is of sum = sum of derivations

$$\frac{\partial J}{\partial \theta_j} = \frac{\partial}{\partial \theta_j} \frac{1}{2}(h_\theta - y)^2 = (h_\theta(x) - y) \cdot \frac{\partial h_\theta(x)}{\partial \theta_j} = (h_\theta(x) - y)\, x_j$$

$h_\theta = \theta_0 x_0 + \theta_1 x_1 + \dots$

$$\Rightarrow \theta_j := \theta_j - \underbrace{\alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})\, x_j^{(i)}}_{= \frac{\partial J}{\partial \theta_j}}$$

← when remembering that we're dealing with a sum here, in J.

keep in mind that in practice, it's always good to normalize J in some way, e.g. by dividing it by m (averaging).

Repeat the descent until convergence, for $j = 0, 1, 2, \dots n$

(normal) Batch Gradient descent: computes gradient exactly per iteration

not easy when m is huge. e.g. hundred million ($m \sim 10^8$)

Repeat {

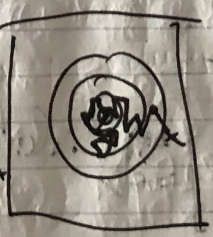
For i = 1 to m {

$\theta_j := \theta_j - \alpha (h_\theta(x^{(i)}) - y^{(i)})\, x_j$ ← data is shuffled

}

}

Stochastic gradient descent: updates $\vec{\theta}$ using one example at a time (or a mini-batch). Sample θ_1 is stochastically chosen



so noisy & random path, but on avg. moves towards minimum.

θ_0

Normal Equation: Gives result immediately (solve linear problem)

$\theta \in \mathbb{R}^{n+1}$

$$\nabla_\theta J(\theta) = \begin{bmatrix} \partial J/\partial \theta_0 \\ \partial J/\partial \theta_1 \\ \vdots \\ \partial J/\partial \theta_n \end{bmatrix}$$

Let's say $A \in \mathbb{R}^{2\times 2} \Rightarrow A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$, $f: \mathbb{R}^{2\times 2} \to \mathbb{R}$

e.g. $f(A) = A_{11} + A_{12}^2 \Rightarrow f\left(\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}\right) = 5 + 6^2 = 41$

$$\nabla_A f(A) = \begin{bmatrix} \partial f/\partial A_{11} & \partial f/\partial A_{12} \\ \partial f/\partial A_{21} & \partial f/\partial A_{22} \end{bmatrix}$$

For the above example, it'd be $\nabla_A f(A) = \begin{bmatrix} 1 & 2A_{12} \\ 0 & 0 \end{bmatrix}$

$\nabla_\theta J(\theta) \overset{\text{set}}{=} \vec{0}$ Gives solution to optimization problem.

If A is square ($A \in \mathbb{R}^{n \times n}$), then $\operatorname{tr} A = \sum_i A_{ii}$

Interesting note: If $f(A) = \operatorname{tr} AB \Rightarrow \nabla_A f(A) = B^T$. $\operatorname{tr} ABC = \operatorname{tr} CAB$. $\nabla_A \operatorname{tr} A A^T C = CA + C^T A$. (like $\frac{\partial}{\partial a} a^2 b = 2ab$)

$$J(\theta) = \tfrac{1}{2} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2; \quad X\theta = \begin{bmatrix} -(x^{(1)})^T- \\ -(x^{(2)})^T- \\ \vdots \\ -(x^{(m)})^T- \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}$$

$$= \begin{bmatrix} x^{(1)T}\theta \\ x^{(2)T}\theta \\ \vdots \\ x^{(m)T}\theta \end{bmatrix} = \begin{bmatrix} h_\theta(x^{(1)}) \\ h_\theta(x^{(2)}) \\ \vdots \\ h_\theta(x^{(m)}) \end{bmatrix}. \text{ and } \vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

$\Rightarrow J(\theta) = \frac{1}{2}(X\theta - y)^T(X\theta - y)$ $\Leftrightarrow$ sum of squared errors.

$$X\theta - y = \begin{bmatrix} h_\theta(x^{(1)}) - y_1 \\ \vdots \\ h_\theta(x^{(m)}) - y_m \end{bmatrix}$$

$$\nabla_\theta J(\theta) = \nabla_\theta \tfrac{1}{2}(X\theta - y)^T(X\theta - y) = \tfrac{1}{2}\nabla_\theta(\theta^T X^T - y^T)(X\theta - y)$$

$$= \tfrac{1}{2}\nabla_\theta [\theta^T X^T X\theta - \theta^T X^T y - y^T X\theta + y^T y]$$

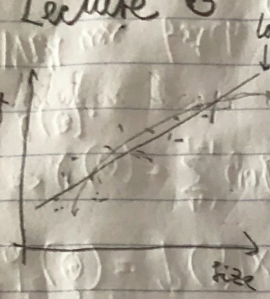
[c] $(ax-b)(ax-b) = a^2x^2 - axb - bax + b^2$

$$= \tfrac{1}{2}[X^T X\theta + X^T X\theta - X^T y - X^T y] = X^T X\theta - X^T y \overset{\text{set}}{=} \vec{0}$$

$X^T X\theta = X^T y$ "normal equations" $\Rightarrow \underline{\theta_{\text{opt.}} = (X^T X)^{-1} X^T y}$

Note: If $X^T X$ is singular, you probably have redundant features. Try to while you could simply proceed with a pseudoinverse, best is to go into the features and figure out what's wrong.

Lecture 3

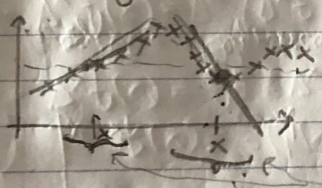


Can apply machinery of lin. reg., but with transformed features $x_1=x$ & $x_2=x^2$ for ex.

Locally Weighted Regression

"Parametric" learning algo: Fit fixed set of parameters (θ_i) to data. Eg lin. reg. (=LR)

"Non-parametric" learning algo: Amount of data/parameters you need to keep grows (linearly) with size of data.



To evaluate h at certain x:

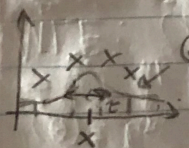
LR: Fit θ to minimise $J(\theta)=\frac{1}{2}\sum_i [y^{(i)}-\theta^T x^{(i)}]^2$
& return $\theta^T x$.

LWR: Fit θ to minimise $\sum_i w^{(i)}(y^{(i)}-\theta^T x^{(i)})^2$
where $w^{(i)}$ is a "weighting" function

$$w^{(i)}=\exp\left[\frac{-(x^{(i)}-x)^2}{2\tau^2}\right]$$ ← tells you how much attention to pay to $(x^{(i)},y^{(i)})$

If $|x^{(i)}-x|$ is small, $w^{(i)}\approx 1$.
If $|x^{(i)}-x|$ is large, $w^{(i)}\approx 0$.

Basically, LWR prioritises accuracy close to the x where you want to make your prediction.



τ = "bandwidth" ← has an effect on over- & underfitting

"Not great at extrapolation"

Why least squares?

Assume $y^{(i)}=\theta^T x^{(i)}+\varepsilon^{(i)}$ ← "Error", aka "unmodelled effects/random noise"

and $\varepsilon^{(i)}\sim\mathcal{N}(0,\sigma^2)$, IID (big assumption). $\Rightarrow P(\varepsilon^{(i)})=\frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{\varepsilon^{(i)2}}{2\sigma^2}\right]$

$$\Rightarrow P(y^{(i)}|x^{(i)};\theta)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{[y^{(i)}-\theta^T x^{(i)}]^2}{2\sigma^2}} \Leftrightarrow [y^{(i)}|x^{(i)};\theta]\sim\mathcal{N}(\theta^T x^{(i)},\sigma^2)$$

← "parameterised eg"

Least squares falls naturally out of these assumptions.

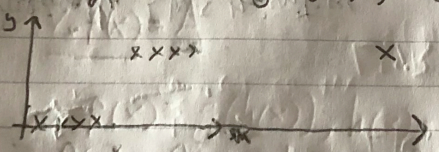
"Likelihood of θ" $L(\theta) = p(\vec{y}|X;\theta) = \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$

L's Dual interpretation: fixed data = likelihood of θ; fixed θ = probability of data

"Log-likelihood" $\ell(\theta) = \log L(\theta) = \log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp(\dots) = m \log \frac{1}{\sqrt{2\pi}\sigma} + \sum_{i=1}^{m} -\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}$

MLE: Choose θ to maximise $L(\theta)$ / $\log L(\theta)$, i.e choose θ to minimise $\sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2 = J(\theta)$ $\Rightarrow$ Choosing θ to minimise J is equivalent to maximising probability of data given iid + Normal assumptions.

Classification: $y \in \{0,1\}$ (binary classification)

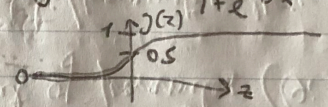


× Linear regression is not a good classification algorithm.

Logistic Regression: most commonly used algorithm.

Want $h_\theta(x) \in [0,1]$. $h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$, $g(z) = \frac{1}{1+e^{-z}}$

= "Sigmoid" or "logistic function".



Tumor malignant: $y \in \{0,1\}$

$P(y=1|x;\theta) = h_\theta(x)$
$P(y=0|x;\theta) = 1 - h_\theta(x)$
$\Rightarrow P(y|x;\theta) = h(x)^y (1-h(x))^{1-y}$

$L(\theta) = P(\vec{y}|X;\theta) = \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta) = \prod_{i=1}^{m} h_\theta(x^{(i)})^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}}$

$\Rightarrow \ell(\theta) = \sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1 - h_\theta(x^{(i)}))$. Need to find θ that maximises $\ell(\theta)$.

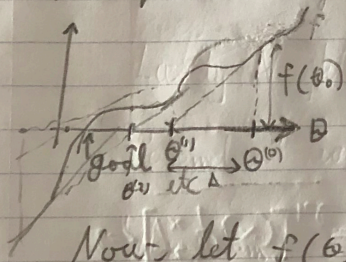
Will use batch gradient ascent: $\theta_j := \theta_j + \alpha \frac{\partial \ell(\theta)}{\partial \theta_j}$

$$\Rightarrow \theta_j := \theta_j + \alpha \underbrace{\left[\sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}\right]}_{= \frac{\partial \ell}{\partial \theta_j}}$$

(PS: logistic regression w/ the sigmoid is concave optimization! Very nice!)

Newton's Method:

Have f. Want to find θ s.t. $f(\theta) = 0$. [or Want to maximize $\ell(\theta)$, i.e. want $\ell'(\theta) = 0$]



$$\theta^{(1)} := \theta^{(0)} - \Delta; \quad f'(\theta^{(0)}) = \frac{f(\theta^{(0)})}{\Delta} \Rightarrow \Delta = \frac{f(\theta^{(0)})}{f'(\theta^{(0)})}$$

$$\Rightarrow \theta^{(t+1)} := \theta^{(t)} - \frac{f(\theta^{(t)})}{f'(\theta^{(t)})}$$

Now, let $f(\theta) = \ell'(\theta) \Rightarrow \theta^{t+1} := \theta^t - \frac{\ell'(\theta^t)}{\ell''(\theta^t)}$

Newton's method is fast. "Quadratic convergence" $\Rightarrow$ error $\propto \epsilon^{-2}$ for smooth functions

When θ is a vector: $\theta^{t+1} := \theta^t + \underbrace{H^{-1}}_{\mathbb{R}^{n+1 \times n+1}} \underbrace{\nabla_\theta \ell}_{\text{vector } \mathbb{R}^{n+1}}$, where H is the Hessian.

$H_{ij} \equiv \frac{\partial^2 \ell}{\partial \theta_i \partial \theta_j}$ ← expensive to (must) compute for many params.

$\Rightarrow$ If # of params not too much, use Newton's method. Otherwise, gradient descent. Or just use quasi-newton...

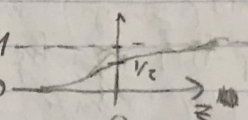
[Very messy temp-lecture]

Lecture 4

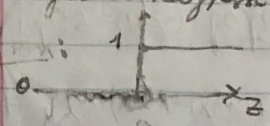
Topics: Perceptron, Exponential family, Generalised Linear Models, Softmax Regression (Multiclass classification).

Perceptron

Logistic Regression: sigmoid function



Perceptron:



$g_p(z) = \begin{cases} 1, z \geq 0 \\ 0, z < 0 \end{cases}$, $h_\theta(x) = g_p(\theta^T x)$

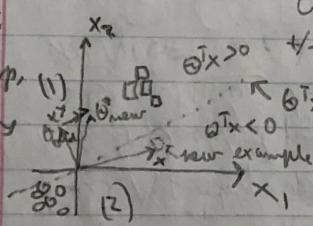
gradient of relevant CF?

$\theta_j := \theta_j + \alpha \underbrace{(y^i - h_\theta(x^i))}_{\text{scalar}} x_j^i$ ← Perceptron update?

$0 \rightarrow$ algorithm got it right

$+/-1 \rightarrow 1$ if wrong & $y^i = 1$; -1 if wrong & $y^i = 0$.

Not used in practice: this probabilistic interp, and other limitations



(actually, $\theta^T x$)

We want: $\theta^T x / y=1$ and $\theta^T x / y=0$

$\Rightarrow$ Just move θ towards x when $y=1$, or move away when $y=0$.

Exponential families

Def: PDF can be written in form: $p(y;\eta) = b(y) \exp[\eta^T T(y) - a(\eta)]$

y - Data. η - natural parameter. $T(y)$ - sufficient statistic $= y$ for today.

$b(y)$ - Base measure; $a(\eta)$ - Log-partition (log, because $p(y;\eta) = \frac{b(y)\exp(\eta^T T(y))}{\exp(a(\eta))}$)

Ex: Bernoulli (Binary data): ϕ = probability of event. Is it part of "expon. families?

("mathematical if-then")

$p(y;\phi) = \phi^y (1-\phi)^{1-y} = \exp\left(\log(\phi^y(1-\phi)^{1-y})\right) = \exp\left[\underbrace{\log\frac{\phi}{1-\phi}}_{\eta} \underbrace{y}_{T(y)} + \underbrace{\log(1-\phi)}_{a(\eta)}\right]$.

From this, we see that: $b(y)=1$ & $T(y)=y$ & $\eta = \log\frac{\phi}{1-\phi} \Leftrightarrow \phi = \frac{e^{-\eta}}{1+e^{-\eta}}$ & $a(\eta) = \log(1+e^{\eta})$

Ex: Gaussian (w/ fixed variance):

Let $\sigma^2 = 1$. $p(y;\mu) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(y-\mu)^2}{2}\right) = \underbrace{\frac{1}{\sqrt{2\pi}}e^{-y^2/2}}_{b(y)} \exp\left[\underbrace{\mu}_{\eta} \underbrace{y}_{T(y)} - \underbrace{\mu^2/2}_{a(\eta)}\right]$

$b(y) = \frac{1}{\sqrt{2\pi}}\exp(-y^2/2)$; $T(y)=y$, $\eta=\mu$, $a(\eta) = \mu^2/2$

When to use what model, I think is →

Real value data: Gaussian
Binary-valued: Bernoulli
Count: Poisson
$\mathbb{R}^+$: Gramma/exponential

Distr over distr: Beta, Dirichlet & Bayesian

why we like exponFams

Propaties

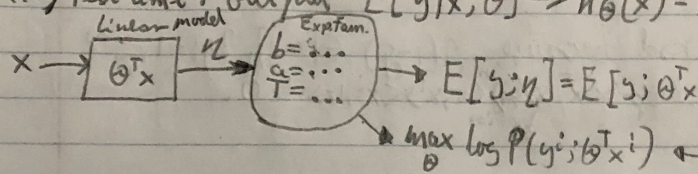
a) MLE wrt η is concave / NLL is convex
b) $E[Y;\eta] = \partial a(\eta)/\partial\eta$
c) $\mathrm{Var}[Y;\eta]$ $\partial^2 a(\eta)/\partial\eta^2$

} no integration!:)

Generalised Linear models (GLM)

Assumptions/design choices:

i) $y|x;\theta \sim$ Exponential Family (η)
ii) $\eta = \theta^T x$, $\theta \in \mathbb{R}^n$, $x \in \mathbb{R}^n$
iii) Test time: output $E[y|x;\theta] \Rightarrow h_\theta(x) = E[y|x;\theta]$



GLM Training

Learning update rule: $\theta_j := \theta_j + \alpha(y^i - h_\theta(x^i))x_j^i$ ← this apparently generically falls out when taking gradients etc.

Terminology η – "natural parameter"; $E(Y;\eta) = g(\eta)$ → "canonical response func."
$\eta = g^{-1}[E(Y;\eta)]$ – "link function". [also, from exp. b: $g(\eta) = \partial_\eta a$]

Parameter names

θ – model param., η – natural param. Canonical params: ϕ-Bernoulli, $\mu\sigma^2$-Gauss, λ-Poisson

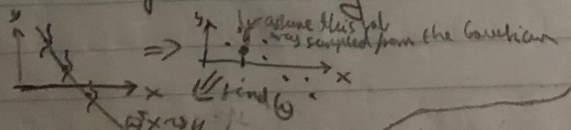
learn – $\theta^T x$ → Design choice

g, g^{-1}

Logistic regression $h_\theta(x) = E(y|x;\theta) = \phi = \frac{1}{1+e^{-\eta}} = \frac{1}{1+e^{-\theta^T x}}$ ← sigmoid pops out naturally from this machinery

Note: Task decides your choice of distribution.

Assumptions Regression etc..

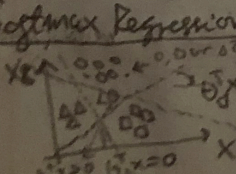


K – # classes; $x^i \in \mathbb{R}^n$; label y
$= \{0,1\}^K$ e.g. $[0,0,1,0]$ ← ○ △ ☆

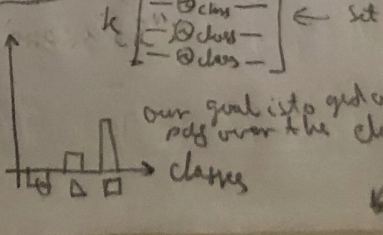
$\theta_{class} \in \mathbb{R}^n$; K such class $\in \{\square, \triangle, \ldots\}$

$K \begin{bmatrix} -\theta_{class}- \\ -\theta_{class}- \\ -\theta_{class}- \end{bmatrix}$ ← set of params per class

Softmax Regression: Cross-entropy



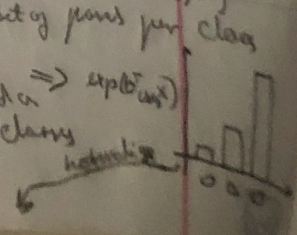
Given x
$\theta_{class}^T x$
logit



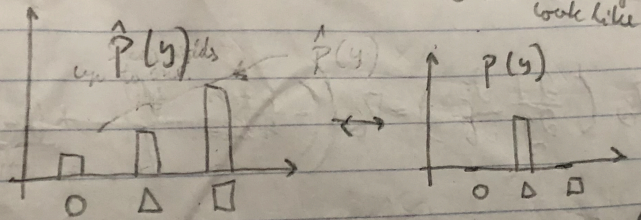
$\Rightarrow \exp(\theta_{class}^T x)$

our goal is to get a pdf over the classes

normalise



$$\frac{e^{\theta_{class}^T x}}{\sum_{i \in \{\circ, \triangle, \square\}} e^{\theta_i^T x}}$$



Goal: make left distr look like right

$$\text{Cross Ent}(p, \hat{p}) = \sum_{s \in \{\circ, \triangle, \square\}} p(y) \log \hat{p}(y) = -\log \hat{P}(y_0) = -\log \frac{e^{\theta_{y_0}^T x}}{\sum_{c \in \{\circ, \triangle, \square\}} e^{\theta_c^T x}}$$

treat as loss & do gradient descent.