Lecture 9 (Indian)

Setup/Assumptions
Bias/Variance
Approx Estim
Empirical risk minimizer
Uniform Convergence
VC dimension

Assumptions

1. Data distribution D: $(x, y) \sim D$ — train / test
det. func.

2. Independent samples
θ^* or h^* = "True params",
not random

$\{x^1, y^1 \dots x^m, y^m\} \sim D$ → learning algo → Estimator

→ S

random variables

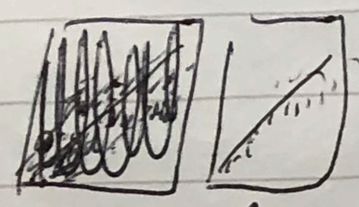
$\hat{h}$ or $\hat{\theta}$

↑ random variables

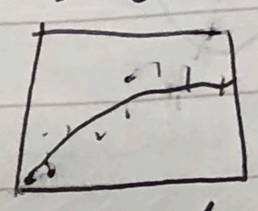
↑ 2 sampling distribution

Data view

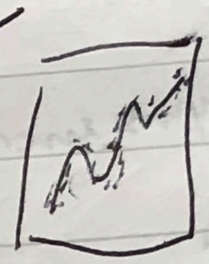
Reg



underfit
(High bias)



correct

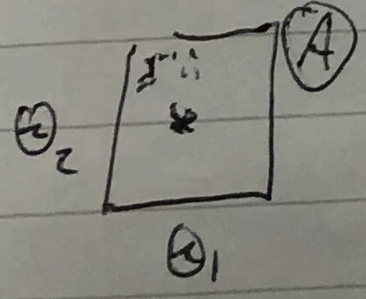


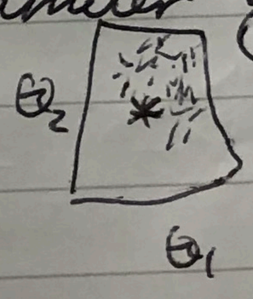
overfit
(High variance)

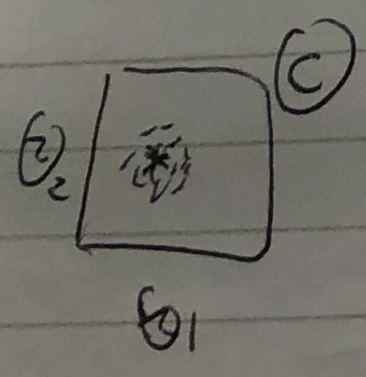
Parameter view

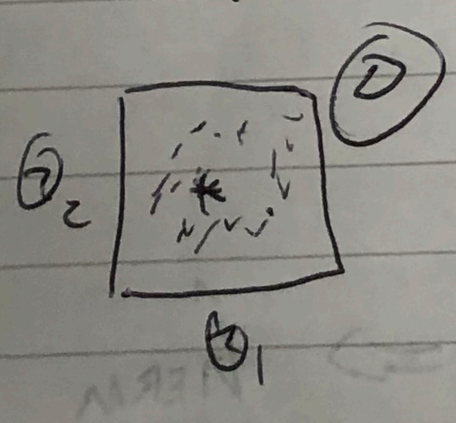
algorithm : m datapoints

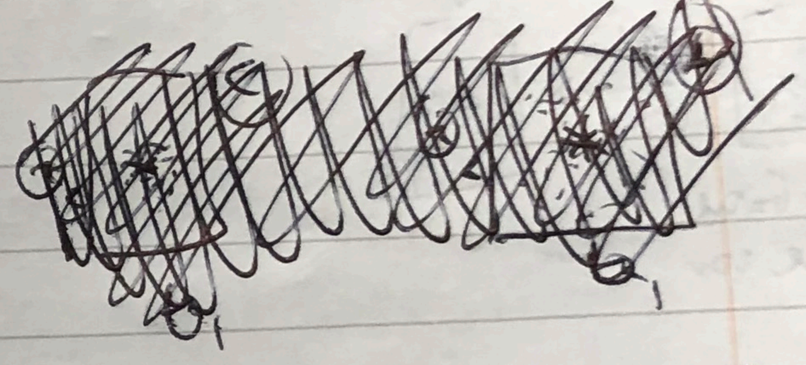
bias







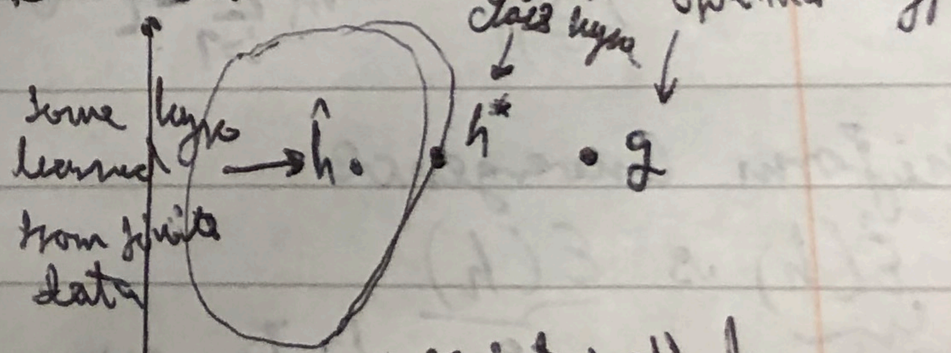




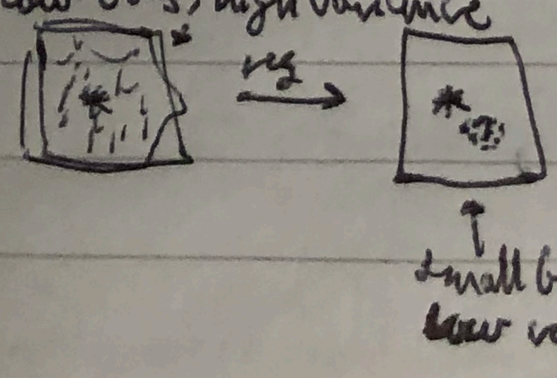
⇒ Bias: is sampling distribution centered around "true para"?
Variance: var. of sampling distr

variance
$Var[\hat{\theta}]$

As $m \to \infty$, variance $\to 0$. The rate of this = "statistical efficiency".
$\hat{\theta} \to \theta^*$ as $m \to \infty$: Consistent algo. ⇒ $E(\hat{\theta}) = \theta^*$



Fighting variance:
1) $m \to \infty$
2) Regularisation:



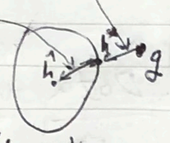
$\varepsilon(h)$ = Risk/generalisation error = $E_{(x,y)\sim D}[\mathbb{1}\{h(x)\neq y\}]$ (in case of classification)

$\hat{\varepsilon}_S(h)$: empirical risk = $\frac{1}{m}\sum_{i=1}^{m}\mathbb{1}\{h(x^i)\neq y^i\}$, S = sample of the m datapoints

$\varepsilon(g)$ = Bayes error / irreducible error

$\varepsilon(h^*) - \varepsilon(g)$ = Approximation error } class

$\varepsilon(\hat{h}) - \varepsilon(h^*)$ = Estimation error:

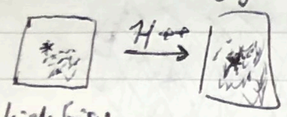


$\varepsilon(\hat{h})$ = ~~Estimation error~~ + Approx error + Irreducible error

est var est bias

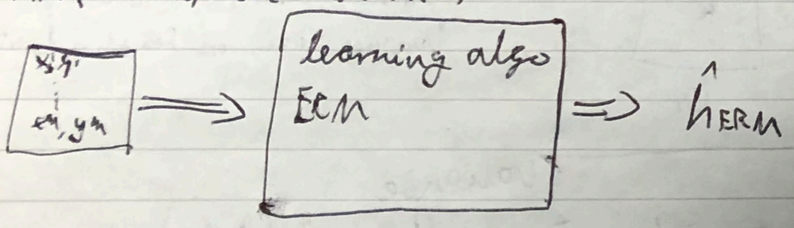
Fight high bias ideas:

1. Make $\mathcal{H}$ bigger



high bias & some var

ERM (emp. risk minimizer)



$$\hat{h}_{ERM} = \underset{h\in H}{\operatorname{argmin}} \frac{1}{m}\sum_{i=1}^{m}\mathbb{1}\{h(x^i)\neq y^i\}$$

Uniform Convergence

1) $\hat{\varepsilon}(h)$ vs $\varepsilon(h)$
(training err. / generalisation err.)

2) $\varepsilon(\hat{h})$ vs $\varepsilon(h^*)$

Bloody lecturer...

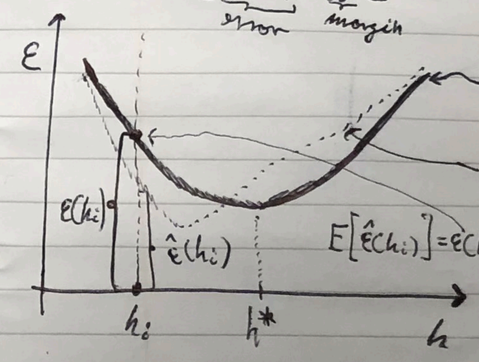
Tools

1) Union bound $A_1, A_2, \dots A_k$ (need not be indep.)

$$P(A_1 \cup A_2 \dots \cup A_k) \leq P(A_1) + P(A_2) + \dots + P(A_k)$$

2) Hoeffding's inequality

Let $z_1, z_2, \dots, z_m \sim \text{Bern}(\phi)$; $\hat{\phi} = \frac{1}{m}\sum_{i=1}^{m} z_i$, and $\gamma > 0$ (margin).
Then, $P_r[|\hat{\phi} - \phi| > \gamma] \leq 2\exp(-2\gamma^2 m)$
($|\hat{\phi} - \phi|$: error; γ: margin)



Hoeffding's ineq.: $Pr[|\hat{\varepsilon}(h_i) - \varepsilon(h_i)| > \gamma] \leq 2\exp(-2\gamma^2 m)$

Finite hypothesis class $\mathcal{H}$

? → $|H| = k$; $Pr[\exists h \in H \,|\, \hat{\varepsilon}_S(h) - \varepsilon(h)| > \gamma] \leq k\,2\exp(-2\gamma^2 m)$

? → $\Rightarrow P[\forall h \in H \,|\, \hat{\varepsilon}(h) - \varepsilon(h)| < \gamma] \geq 1 - \underbrace{2k\exp(-2\gamma^2 m)}_{\delta}$

Let $\delta = 2k\exp(-2\gamma^2 m)$; δ - prob of error; γ - margin of error; m - sample size.

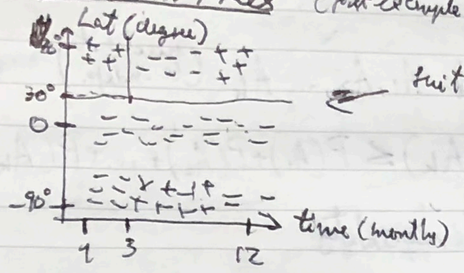
$\Rightarrow$ Can solve for δ, γ, m when knowing the others. Eg. fix $\gamma, \delta > 0$, $m \geq \frac{1}{2\gamma^2}\log\frac{2k}{\delta}$. Known as the "sample complexity result".

VC dimension: $VC(H) = \dots$

$\Rightarrow \varepsilon(\hat{h}) \leq \varepsilon(h^*) + O\left[\sqrt{\frac{VC(H)}{m}\log\left[\frac{m}{VC(H)} + \frac{1}{m}\log\frac{1}{\delta}\right]}\right]$ — relates to the above

Lecture 10
- Decision trees
- Ensemble methods
- Bagging
- Random forests
- Boosting

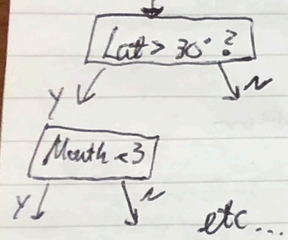
Decision Trees (first example on nonlinear model)



← suitability for skiing

Greedy, top-down, recursive

Partitioning



etc...

⇒ Region R_p

Looking for a split S_p

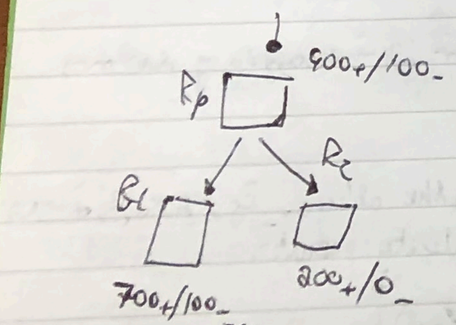
$S_p(j,t) = (\{X | X_j < t, X \in R_p\}, \{X | X_j > t, X \in R_p\})$ (R_1, R_2)

How to choose splits?

Define $L(R)$: loss on R. Given C classes, define $\hat{P}_c$ to be proportion of examples in R that are of class C.

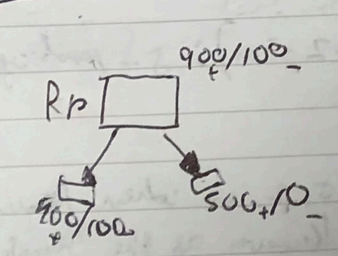
$\Rightarrow L_{misclass} = 1 - \max_c \hat{P}_c$. $\min_{j,t}$ $\underbrace{L(R_p)}_{\text{parent loss}} - \underbrace{\left(\frac{|R_1|}{|R_1|+|R_2|} L(R_1) + \frac{|R_2|}{|R_1|+|R_2|} L(R_2)\right)}_{\text{children loss}}$ ← ?

Misclassification Loss has issues!



$\frac{8}{10} L(R_1) + \frac{2}{10} L(R_2)$

$= \frac{8}{10} \frac{1}{8} + \frac{2}{10} 0 = \frac{1}{10}$



$L(R_1) + L(R_2)$

$= \frac{1}{2} \frac{1}{5} + \frac{1}{2} 0$

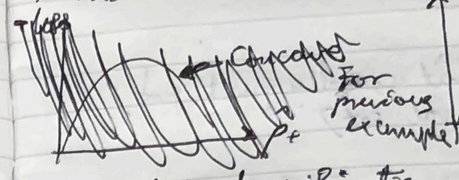
$= \frac{1}{10}$

$L(R_p) = \frac{1}{10}$.

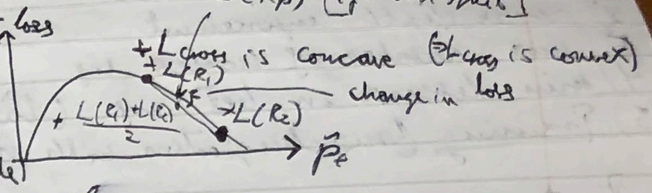
⇒ Not sensitive enough, as right example is clearly preferable.

Instead, define cross-entropy loss

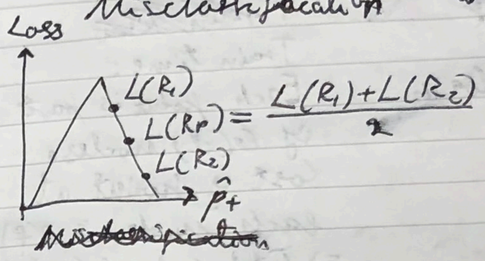
$L_{cross} = -\sum_c \hat{p}_c \log \hat{p}_c$



$+ L(R_p)$ [if even split]



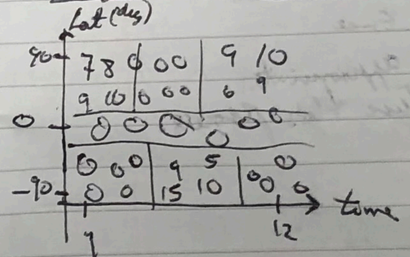
Misclassification



~~Misclassification~~

Gini: $\sum_c \hat{P}_c (1 - \hat{P}_c)$. Looks similar to cross-entropy one.

Regression trees

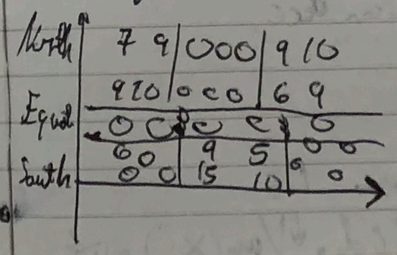


R_m

Predict $\hat{Y}_m = \frac{\sum_{i \in R_m} (Y_i - \hat{Y}_m)}{|R_m|}$

$L_{squared} = \sum_{i \in R_m} \frac{(Y_i - \hat{Y}_m)^2}{|R_m|}$

Categorical Variables



"Loc ∈ {Northern}?"

7 categories

⇓

2^7 possible splits

← Is binary?

Regularisation of DTs

1) min leaf size enforcement
2) max depth enforcement
3) max number of nodes enf.
4) min decrease in loss enf. (but usually isn't good)
5) Pruning (misclassifications with val. set)

Runtime:

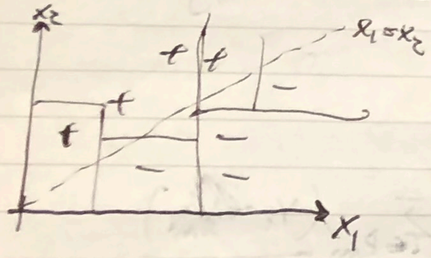
n examples
f features
d depth

~~Test time:~~
Test time:
$O(d)$; d depth

~~Train time:~~
Train time:
Each point is part of $O(d)$ nodes.
Cost of point at each node is $O(f)$
$\Rightarrow$ Total cost = $O(nfd)$

Note that Datamatrix is of size nf.

No additive structure



Compare to e.g. linreg. to decision trees here... dec. tree doesn't have the appropriate structure to capture this process.

Recap

+ Easy to explain
+ Interpretable
+ Categorical vars
+ Fast

- High variance
- Bad at additive
- Low precision etc.

Ensembling

Take X_is which are RVs that are ~~indep~~ iid. Let $\text{Var}(X_i) = \sigma^2$.
$\Rightarrow \text{Var}(\bar{X}) = \text{Var}\left(\frac{1}{n}\sum_i X_i\right) = \sigma^2/n$.

If we drop independence assumption so that X_is are now id: X_is correlated by ρ.

$\Rightarrow \text{Var}\,\bar{X}_B = \rho\sigma^2 + \frac{1-\rho}{n}\sigma^2$

Ways to ensemble (for reducing variance)

1) Different algorithms
2) Different training sets
3) Bagging (eg Random forests)
4) Boosting (eg Adaboost, xgboost)

Bagging: Bootstrap aggregation

Have a true population p
Training set $S \sim p$
Assume $p = S$

Bootstrap samples $Z \sim S$ (sample with replacement)

Eg $Z_1, \ldots, Z_M$. Now train model G_m on Z_m.

$$G = \frac{\sum_{m=1}^{M} G_m}{M}$$

Bias-variance Analysis of this: $\text{Var}(\bar{x}) = \rho\sigma^2 + \frac{1-\rho}{M}\sigma^2$

Bootstrapping is driving down ρ. More $M \to$ less variance
Bias slightly increases because of random subsampling

DTs + Bagging

DTs are high variance / low bias ⇒ Making this an ideal fit for bagging.

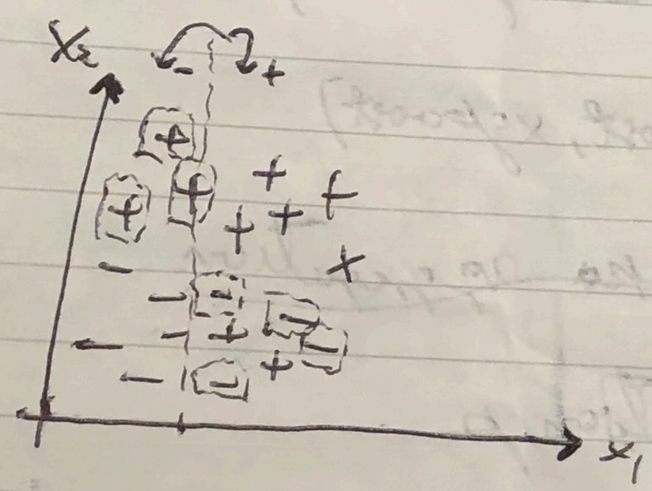
RFs

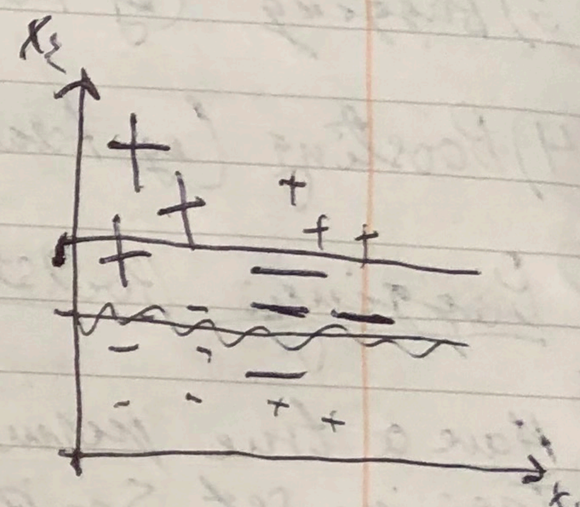
At each split, consider only a fraction of your total features:

This decreases ρ, because otherwise all individual trees would find each esp splitting at same "high value" feature.
→ Decorrelates your models

Boosting

Decrease bias
Additive





Determine for classifier G_m a weight α_m proptn.

$\log\left(\frac{1-err_m}{err_m}\right)$

$G_{(x)} = \sum_m \alpha_m G_m$

Each G_m is trained on reweighted training set. (from step $m-1$?)

Adaboost

2. Better to read on wiki.

Lecture 11

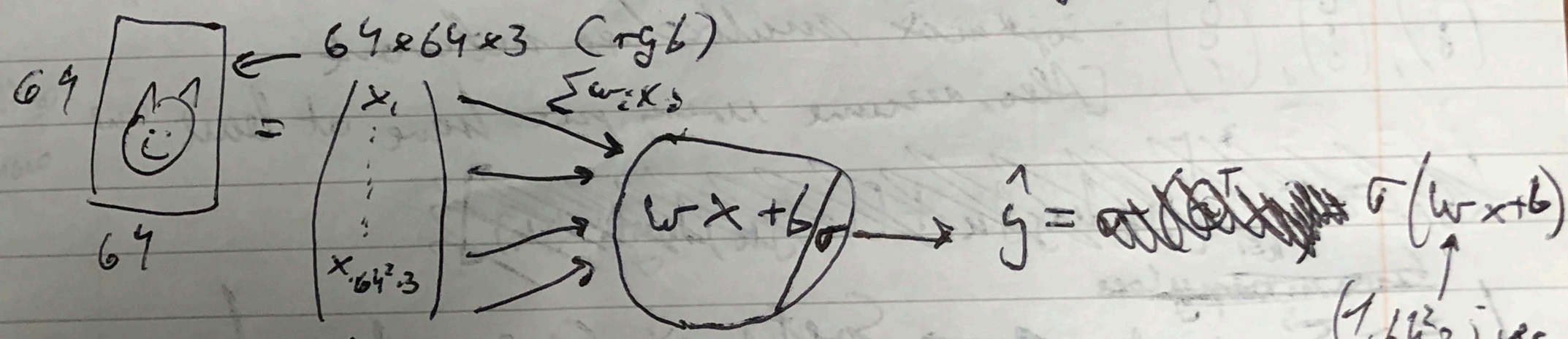
I. Log regr.
II. Neural networks.

Deep learning

- Computational power
- data availability
- algorithms

(I) Logistic regression

Goal: Find cats in images ($1 \rightarrow$ presence of cat, $0 \rightarrow$ absence of cat)

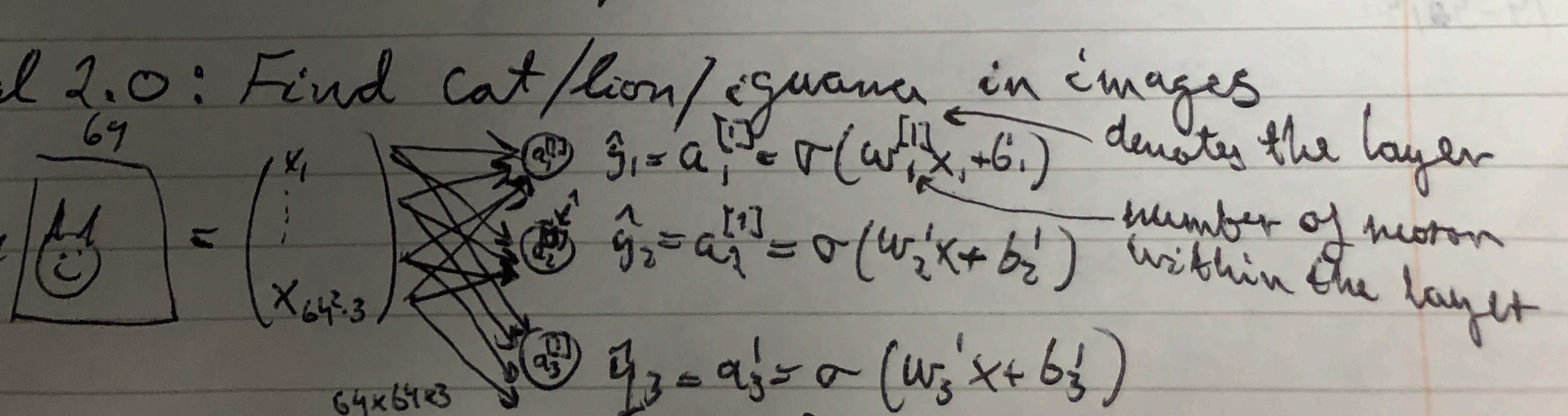


i) initialise w, b
ii) Find the optimal w, b
iii) Use $\hat{y} = \sigma(wx+b)$ to predict

neuron = linear + activation (eq.1)

model = architecture + parameters (eq.2)

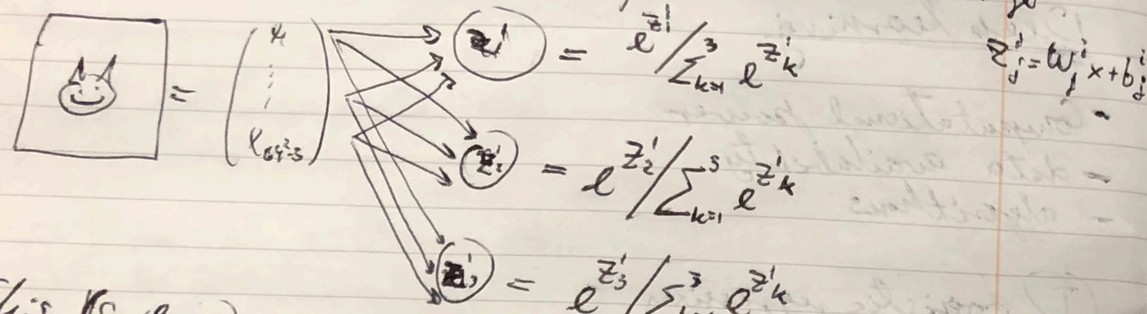
Goal 2.0: Find cat/lion/iguana in images



Needed dataset: Images + labels ~ y

Note:
Neurons independent ⇒ network should be robust to multiple animals (eg cat + lion) in image.

Goal 3.0: + constraint "unique animal on an image"



This (Softmax) scheme is no longer robust to multiple animals in the image, as the denominators must all equal 1.

$\begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\end{pmatrix}$: softmax multiclass network

[Also, assume every pic. have at least one of the animals]

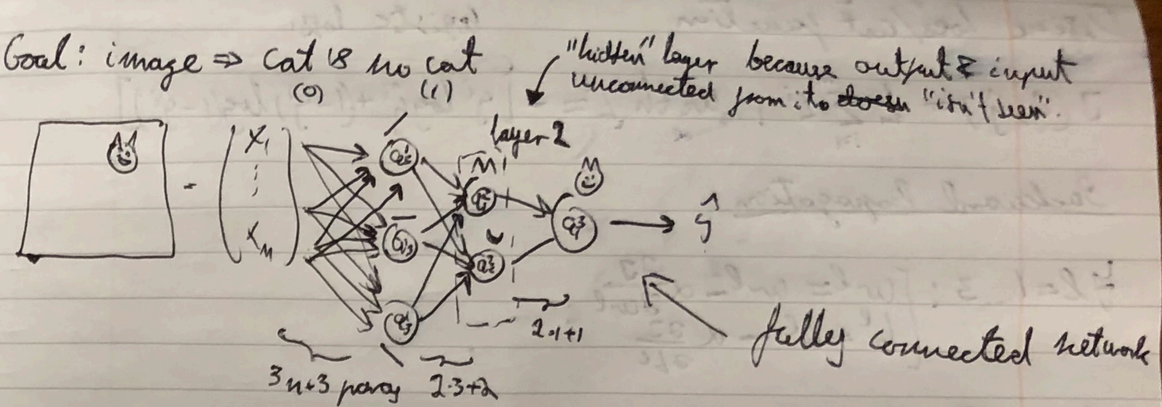
~~$L_{3,1} = -\sum_{k=1}^{3} y_k \log \hat{y}_k$~~

$L_{xEL} = -\sum_{k=1}^{3} y_k \log \hat{y}_k$ ← Softmax cross entropy loss.

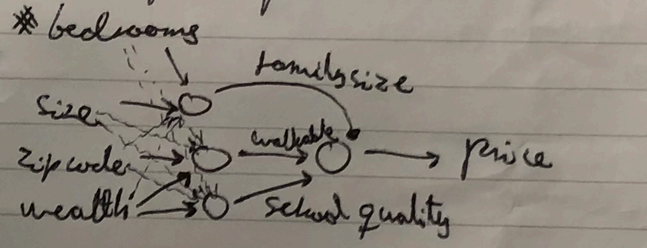
Sidenote: to eg learn the age of a cat, one could do softmax over certain age ranges, or more appropriately perhaps, replace the sigmoid with something like linear regression (but such that the outputted age $\hat{y} \in [0, 35]$ years. Loss function then becomes $|y - \hat{y}|^2$

II Neural Networks

Goal: image ⇒ cat (0) vs no cat (1)



House price prediction



end to end learning / black box model

Propagation equations

$z^1 = w^1 x + b^1$ ~ (3,n)(n,1) + (3,1) = (3,1)
$a^1 = \sigma(z^1)$ ~ (3,1)
$z^2 = w^2 \cdot a^1 + b^2$ ~ (2,3)(3,1) + (2,1) = (2,1)
$a^2 = \sigma(z^2)$ ~ (2,1)
$z^3 = w^3 a^2 + b^3$ ~ (1,2)(2,1) + (1,1) = (1,1)
$a^3 = \sigma(z^3) = \hat{y}$ ~ (1,1)

- What happens for an input batch of m examples

$X = \begin{pmatrix} | & | & & | \\ x^1 & x^2 & \dots & x^m \\ | & | & & | \end{pmatrix}$, (n,m)

$z^1 = w^1 X + b^1$ ~ (3,n)(n,m) + (3,1) broadcasted, i.e. repeated m times = (3,m)

$= \begin{bmatrix} | & | & & | \\ z^{1,1} & \dots & & z^{1,m} \\ | & | & & | \end{bmatrix}$

Optimising weights & intercepts

Define loss/cost function $J(\hat{y}, y)$ (cost) / $\mathcal{L}^{(i)}$ (loss)

$$J(\hat{y}, y) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}^{(i)}, \text{ with } \mathcal{L}^{(i)} = -\overbrace{\left[y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log (1 - \hat{y}^{(i)}) \right]}^{\text{logistic loss}}$$

Backward Propagation

$$\forall l = 1 \dots 3 : \begin{cases} w^{[l]} = w^{[l]} - \alpha \frac{\partial J}{\partial w^{[l]}} \\ b^{[l]} = b^{[l]} - \alpha \frac{\partial J}{\partial b^{[l]}} \end{cases}$$

$$\frac{\partial J}{\partial w^{[3]}} = \frac{\partial J}{\partial a^{[3]}} \frac{\partial a^{[3]}}{\partial z^{[3]}} \frac{\partial z^{[3]}}{\partial w^{[3]}}$$

$$\frac{\partial J}{\partial w^{[2]}} = \frac{\partial J}{\partial z^{[3]}} \frac{\partial z^{[3]}}{\partial a^{[2]}} \frac{\partial a^{[2]}}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial w^{[2]}}$$

$$\frac{\partial J}{\partial w^{[1]}} = \frac{\partial J}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial a^{[1]}} \frac{\partial a^{[1]}}{\partial z^{[1]}} \frac{\partial z^{[1]}}{\partial w^{[1]}}$$

Lecture 12

Outline:

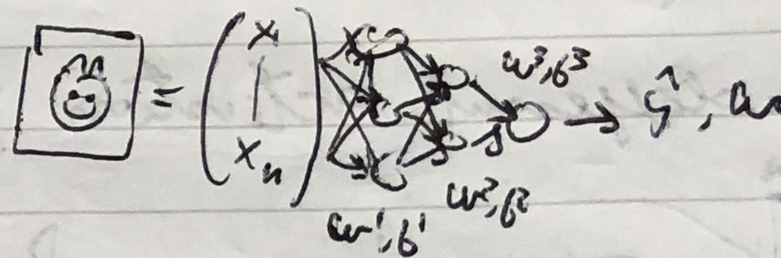
I Logreg from a NN mindset ✓

II Neural Networks
: → backprop

III Improving your NNs

Backprop

Cost function $J(y,\hat{y}) = \frac{1}{m}\sum_{i=1}^{m} L^i(y,\hat{y})$



$$L^i = -\left[y^i \log \hat{y}^i + (1-y^i)\log(1-\hat{y}^i)\right]$$

Update: $w^l := w^l - \alpha \frac{\partial J}{\partial w^l}$

$\partial J/\partial w^3 \Rightarrow \partial L/\partial w^3 = -\left[y^i \frac{1}{\partial w^3}\left(\log \sigma(w^3 a^2 + b^3)\right) + (1-y^i)\frac{1}{\partial w^3}\log\left(1-\sigma(w^3 a^2 + b^3)\right)\right] =$

use that $\sigma'(x) = \sigma(x)(1-\sigma(x))$

$$= -\left[y^i \frac{1}{a^3} a^3(1-a^3)\, a^{2T} + (1-y^i)\frac{1}{1-a^3}\cdot(-1)\, a^3(1-a^3)\, a^{2T}\right]$$

$$= -\left[y^i(1-a^3)a^{2T} - (1-y^i)a^3 a^{2T}\right] = -\left[y^i a^{2T} - a^3 a^{2T}\right]$$

$$= -(y^i - a^3)a^{2T}$$

$\frac{\partial L}{\partial w^3} = \frac{\partial L}{\partial z^3}\frac{\partial z^3}{\partial w^3} \Rightarrow \frac{\partial L}{\partial z^3} = -(y^i - a^3)$

$$\frac{\partial J}{\partial w^3} = \frac{-1}{m}\sum_{i=1}^{m}(y^i - a^3)a^{2T}$$

$= \partial L/\partial z^3$

$$\frac{\partial L}{\partial w^2} = \frac{\partial L}{\partial a^3}\frac{\partial a^3}{\partial z^3}\frac{\partial z^3}{\partial a^2}\frac{\partial a^2}{\partial z^2}\frac{\partial z^2}{\partial w^2}$$

$= (a^3 - y)$ $\qquad a^2(1-a^2)$

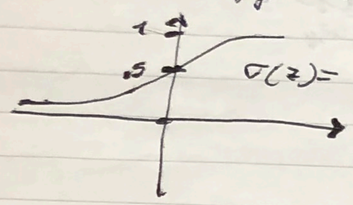
$$\frac{\partial L}{\partial w^2} = \underbrace{(a^3-y)}_{1,1}\underbrace{w^{3T}}_{2,1}\odot\underbrace{a^2(1-a^2)}_{2,1}\underbrace{a^{1T}}_{1,3} = \underbrace{w^{3T}}_{2,1}\odot\underbrace{a^2(1-a^2)}_{2,1}\underbrace{(a^3-y)}_{1,1}\underbrace{a^{1T}}_{1,3}$$

(only possible due to element-wise product)

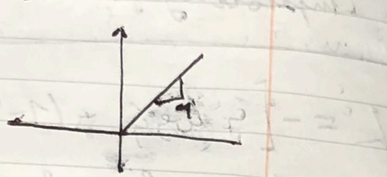
$$\frac{\partial J}{\partial w^2} = \frac{1}{m}\sum_{i=1}^{m} \frac{\partial L}{\partial w^2}$$

III Improving your NNs:

Trick A: Use different activation functions

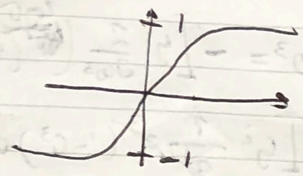


$$ReLU(z) = \begin{cases} 0, & z \leq 0 \\ z, & z > 0 \end{cases}$$



$$ReLU'(z) = 1\{z > 0\}$$

$$\tanh z = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$



$$\tanh'(z) = 1 - \tanh z^2$$

Sigmoid: problem here is that the gradient vanishes at high/low z. Same issue for the tanh

Why do we need activation functions?

activation = Id $z \mapsto z$

$$\hat{y} = a^3 = z^3 = w^3 a^2 + b^3$$

$$= w^3(w^2 a^1 + b^2) + b^3 = w^3 w^2(w^1 x + b^1) + w^3 b^2 + b^3$$

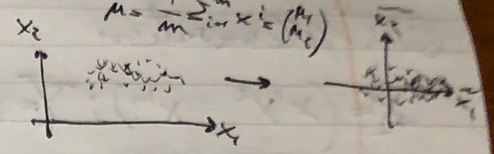
$= Wx + B$, with $W = w^3 w^2 w^1$, $B = w^3 w^2 b^1 + w^3 b^2 + b^3$

⇒ Without activation functions, the NN always becomes simply linear regression.

B Initialisation methodology

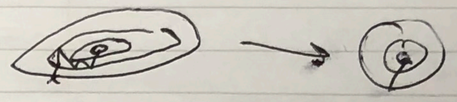
Normalising your input: $X = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^i = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}$$

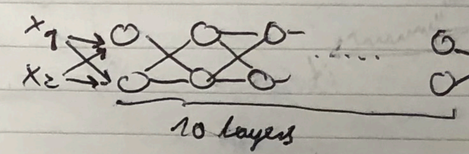


$$\sigma^2 = \frac{1}{m}\sum_i x^{i2} \Rightarrow$$

Effect of normalisation on loss functions:



Vanishing/Exploding gradients

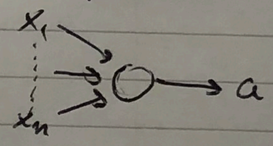


$g: z \mapsto z$
$g = 0$

$$\hat{y} = w^L \cdot a^{L-1} = w^L w^{L-1} \cdot a^{L-2} = \ldots = w^L w^{L-1} \ldots w^1 x$$

$$w^l = \begin{pmatrix} 1.5 & 0 \\ 0 & 1.5 \end{pmatrix} \Rightarrow \hat{y} = \begin{pmatrix} 1.5^L & 0 \\ 0 & 1.5^L \end{pmatrix} x$$

Example with 1 neuron



$a = \sigma(z)$

$z = w_1 x_1 + \ldots + w_n x_n$

⇒ large n requires → small w_i ⇒ $w_i \sim 1/n$

w_i * initialise w/ np.random.randn(shape) * np.sqrt$\left(\frac{1}{n^{L-1}}\right)$ when using Sigmoids. For ReLU: use $2/n^{L-1}$ instead.

Xavier initialisation: $w \sim \sqrt{1/n^{l-1}}$ for tanh

He initialisation: $w^l \sim \sqrt{\frac{2}{n^l + n^{l-1}}}$

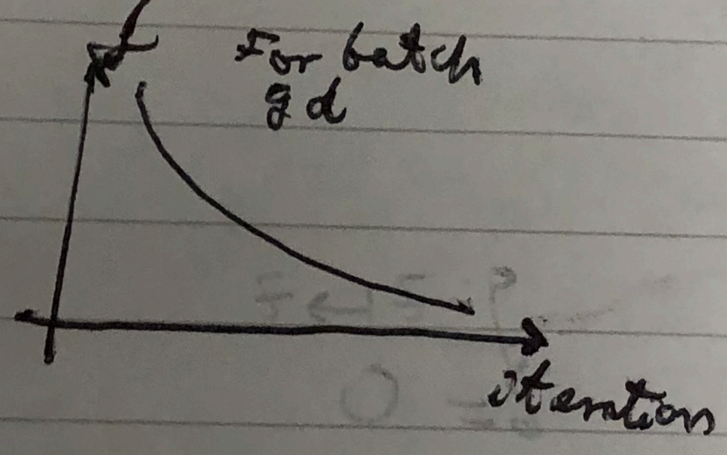
⑥ Optimisation

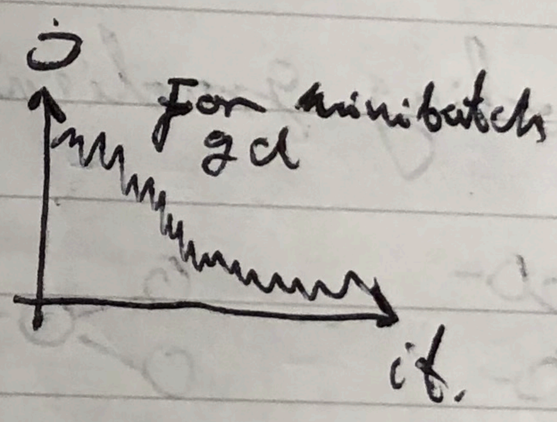
$X = (x^1 x^2 \dots x^m)$; $Y = (y^1 \dots y^m)$; $x = (x^{\{1\}} \dots x^{\{1000\}})$ & $y = (y^{\{1\}} \dots y^{\{1000\}})$

Algo: for iteration $t = 1 \dots$ select batch $(x^{\{t\}}, y^{\{t\}})$, Forward prop backward batch.

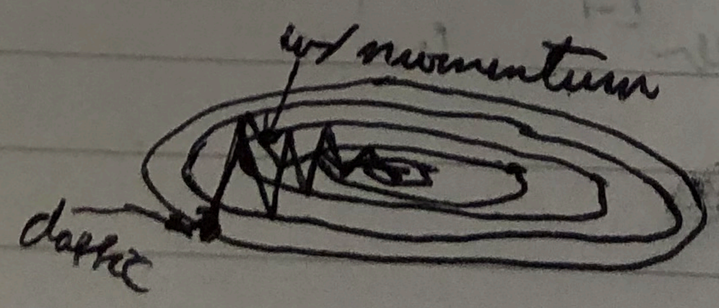
$J = \frac{1}{1000} \sum_{i=1}^{1000} L^i$ update $w^l, b^l \; \forall l$.

graphs:





"GD + momentum" algo:



Closer steeper ↔ smaller, ↔ longer.

$$\begin{cases} v = \beta v + (1-\beta) \frac{\partial L}{\partial w} \\ w = w - \alpha v \end{cases}$$