COST FUNCTION FOR NNs

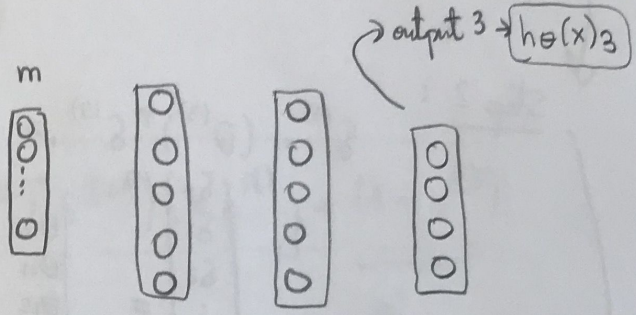
↳ We sum over all output units (instead of only one for LG)

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[y_k^{(i)}\log((h_\Theta(x^{(i)}))_k) + (1-y_k^{(i)})\log(1-(h_\Theta(x^{(i)}))_k)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{j,i}^{(l)})^2$$

↳ Σ over all output units

Reg. term.

- $L \rightarrow$ # of layers
- $s_l \rightarrow$ # of units in layer l
- $K \rightarrow$ # of output units



$\begin{cases} K = 4 \\ L = 4 \end{cases}$ $\begin{cases} S_1 = m \\ S_2 = 5 \\ S_3 = 5 \\ S_4 = K = 4 \end{cases}$

Reg. term:

→ We do not count the bias units

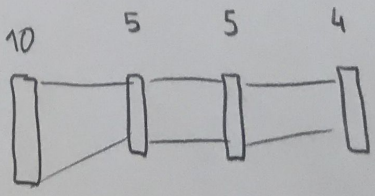
→ In our example: $L = 4$

For $\Theta^{(1)}$: $\rightarrow \sum_{i=1}^{m}\sum_{j=1}^{5}(\Theta_{ji}^{(1)})^2$

$\Theta^{(1)} \rightarrow \begin{bmatrix} \Theta_{11} & \cdots & \\ \vdots & \ddots & \\ \Theta_{5m} & & \end{bmatrix}$ (5 rows, m columns)

$$\Theta_1^{(1)} = \Theta_{10}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \ldots + \Theta_{1m}^{(1)}x_m$$

How many Θ elements to find in the following NN?



$\Theta^{(1)} \rightarrow (10+1)\cdot 5 = 11\cdot 5 = 55$

$\Theta^{(2)} \rightarrow (5+1)\cdot 5 = 6\cdot 5 = 30$

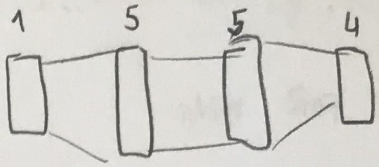
$\Theta^{(3)} \rightarrow (5+1)\cdot 4 = 6\cdot 4 = 24$

} 109 weights / params

(14)

BACKPROP ALGO. FOR NNs

(x, y)



We need: → $J(\Theta)$

→ $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$

For each node we need to compute $\delta_j^{(l)}$ → error of node j in layer l

Step 1: $\boxed{\delta_j^{(4)} = a_j^{(4)} - y_j} = h_\Theta(x)_j - y_j$

↳ $\delta^{(4)} = a^{(4)} - y \longrightarrow \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$

Step 2: $\delta^{(3)} = (\Theta^{(3)})^T \delta^{(4)} .* g'(z^{(3)})$ $\quad \Theta^{(3)} \to 4 \times 6$

↳ $\begin{bmatrix} \delta_0 \\ \delta_1 \\ \delta_2 \\ \vdots \\ \delta_5 \end{bmatrix}^{(3)} = \begin{bmatrix} \Theta_{10} & \cdots & \vdots \\ \Theta_{11} & \ddots & \vdots \\ \Theta_{12} & & \\ & & \\ \Theta_{15} & \cdots & \Theta_{45} \end{bmatrix}^{(3)} \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \end{bmatrix}^{(4)} .* \begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ z_5 \end{bmatrix}^{(3)}$

$6 \times 1 \qquad 6 \times 4 \qquad 4 \times 1 \qquad 6 \times 1$

Step 3: $\delta^{(2)} = (\Theta^{(2)})^T \delta^{(3)} .* g'(z^{(2)})$

ALGO IN DETAIL: training set: $\{(x^{(1)}, y^{(1)}) \dots (x^{(m)}, y^{(m)})\}$

Set $\Delta_{ij}^{(l)} = 0$ for all i, j, l

Loop over training samples $t = 1:m$

① Set $a^{(1)} = x^{(t)}$

② Perform forward prop. to get $a^{(l)}$ for $l = 2:L$

$$\begin{cases} z^{(2)} = \theta^{(1)} a^{(1)} \rightarrow a^{(2)} = g(z^{(2)}) \\ z^{(3)} = \theta^{(2)} a^{(2)} \rightarrow a^{(3)} = g(z^{(3)}) \\ z^{(4)} = \theta^{(3)} a^{(3)} \rightarrow a^{(4)} = h_\theta(x) = g(z^{(4)}) \end{cases}$$

③ Using $y^{(t)}$ → compute $\delta^{(L)} = a^{(L)} - y^{(t)}$

④ Compute $\delta^{(L-1)} \rightarrow \delta^{(2)}$ using $\delta^{(l)} = \left((\theta^{(l)})^T \delta^{(l+1)}\right) .* a^{(l)} .* (1 - a^{(l)})$

g'

$g'(z^{(l)}) = a^{(l)} .* (1 - a^{(l)})$

⑤ $\Delta_{ij}^{(l)} = \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$ ------ vect. ------> $\Delta^{(l)} = \Delta^{(l)} + \delta^{(l+1)} (a^{(l)})^T$

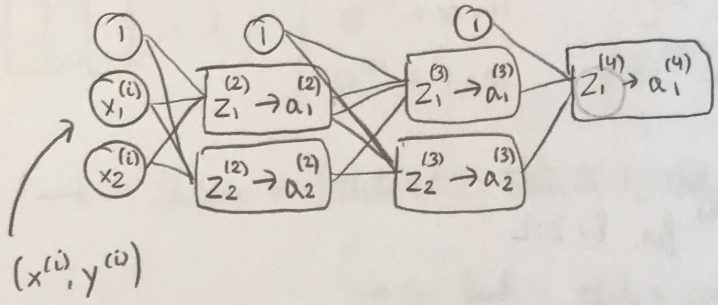
accumulator of our partial der. terms

$$\begin{cases} j \neq 0 \rightarrow D_{ij}^{(l)} = \frac{1}{m} \left(\Delta_{ij}^{(l)} + \lambda \theta_{ij}^{(l)}\right) \\ j = 0 \rightarrow D_{ij}^{(l)} = \frac{1}{m} \Delta_{ij}^{(l)} \end{cases}$$

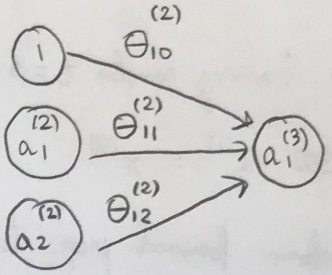
$$\rightarrow \frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta) = D_{ij}^{(l)}$$

BACKPROP. INTUITION

→ Forward prop.



Weights:



$$z_1^{(3)} = \Theta_{10}^{(2)} \cdot 1 + \Theta_{11}^{(2)} \cdot a_1^{(2)} + \Theta_{12}^{(2)} \cdot a_2^{(2)}$$

→ Back. prop.

Special case
- → one example $x^{(i)}, y^{(i)}$
- → 1 output unit
- → $\lambda = 0$

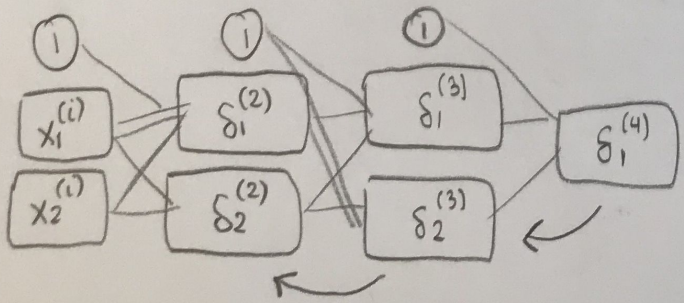
→ $\text{cost}(i) = y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))$

↳ how well is it doing on example (i) ?

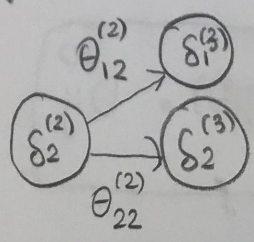
→ $\delta_j^{(l)}$ → error of cost for $a_j^{(l)}$ (unit j in layer l)

↳ $\delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} \text{cost}(i)$

→ measure of how much we would like to change the weights of the NN to affect these values which will affect the output and thus the cost.

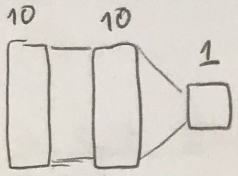


Weights:



$$\delta_2^{(2)} = \Theta_{12}^{(2)} \delta_1^{(3)} + \Theta_{22}^{(2)} \delta_2^{(3)}$$

Unrolling the params (example):



$\Theta^{(1)} \rightarrow 10 \times 11$
$\Theta^{(2)} \rightarrow 10 \times 11$
$\Theta^{(3)} \rightarrow 1 \times 11$

$D^{(1)} \rightarrow 10 \times 11$
$D^{(2)} \rightarrow 10 \times 11$
$D^{(3)} \rightarrow 1 \times 11$

→ `thetavec = [Theta1(:); Theta2(:); Theta3(:)];`

↳ Go back : `Theta1 = reshape(thetavec(1:110), 10, 11);`

In learning algo:

→ Have initial $\Theta^{(1)}, \Theta^{(2)}, \Theta^{(3)}$

→ Unroll for fminunc(@costFunction, initialTheta, options)

→ gradientVec ← costFunction(thetaVec)

→ Get $\Theta^{(1)}, \Theta^{(2)}, \Theta^{(3)}$ from thetaVec (reshape)

→ Compute $D^{(1)}, D^{(2)}, D^{(3)}$

→ Unroll Ds to get gradientVec

Note:

→ Matrix form : better for FP and BP computations

→ Vector form : for functions that assume you have all your params. in one input

(18)

GRADIENT CHECKING

↳ to avoid implementation bugs

Estimation : $\frac{d}{d\theta} J(\theta) \approx \frac{J(\theta+\varepsilon) - J(\theta-\varepsilon)}{2\varepsilon}$ $\quad \varepsilon = 10^{-4}$

For θ vector with multiple params: $\frac{\partial}{\partial \theta_2} J(\theta) \approx \frac{J(\theta_1, \theta_2+\varepsilon, \dots, \theta_n) - J(\theta_1, \theta_2-\varepsilon, \dots, \theta_n)}{2\varepsilon}$

↳ Implement estimation and compare with Ds from backprop.

Random init.

→ Do not init. your $\Theta_{ij}^{(l)}$ to 0s → $a_1^{(2)} = a_2^{(2)}$ → same features learned everywhere

→ Pick random values $[-\varepsilon, \varepsilon]$ → symmetry breaking

OVERALL METHOD:

⓪ Pick a NN architecture

Training :

① Weights init. randomly

② Implement forward prop. → get $h_\Theta(x^{(i)})$ for all $x^{(i)}$

③ Implement cost function

④ Implement BP → get partial derivatives

⑤ Use grad. checking to confirm BP

⑥ Use GD or other to minimise cost function → get $\Theta_{ij}^{(l)}$

EVALUATING A HYPOTHESIS:

→ Split training/test sets
- → Learn θ and minimize $J_{train}(\theta)$ with training set
- → Compute the test set error $J_{test}(\theta)$

→ Test set error:

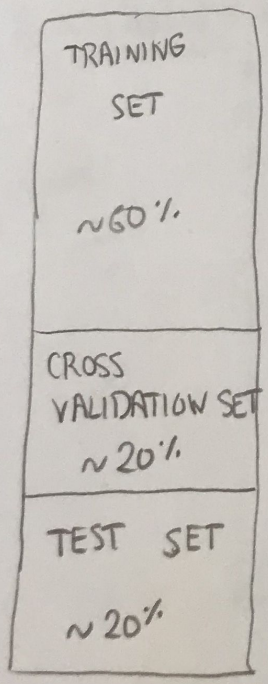
① Lin. reg. →
$$J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} \left(h_\theta(x_{test}^{(i)}) - y_{test}^{(i)}\right)^2$$

② Classification → Misclassification error

$$error(h_\theta(x), y) = \begin{cases} 1 & \text{when } h_\theta(x) \geq 0.5 \text{ and } y=0 \text{ or } h_\theta(x) < 0.5 \text{ and } y=1 \\ 0 & \text{otherwise} \end{cases}$$

→
$$\text{Test Error} = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} err\left(h_\theta(x_{test}^{(i)}), y_{test}^{(i)}\right)$$

MODEL SELECTION:



① Optimize params. in θ by using training set for each model.

② Find the model with smaller error using the CV set.

③ Estimate the generalization error using the test set

} PROCEDURE

* → Usually $J_{CV}(\theta) < J_{test}(\theta)$ because the degree of the polynomial has been fit to the CV set.

(20) Example:

3 models:

① $h_\theta(x) = \theta_0 + \theta_1 x \longrightarrow \theta^{(1)} \longrightarrow J_{CV}(\theta^{(1)})$

② $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 \longrightarrow \theta^{(2)} \longrightarrow J_{CV}(\theta^{(2)})$

③ $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 \longrightarrow \theta^{(3)} \longrightarrow J_{CV}(\theta^{(3)})$

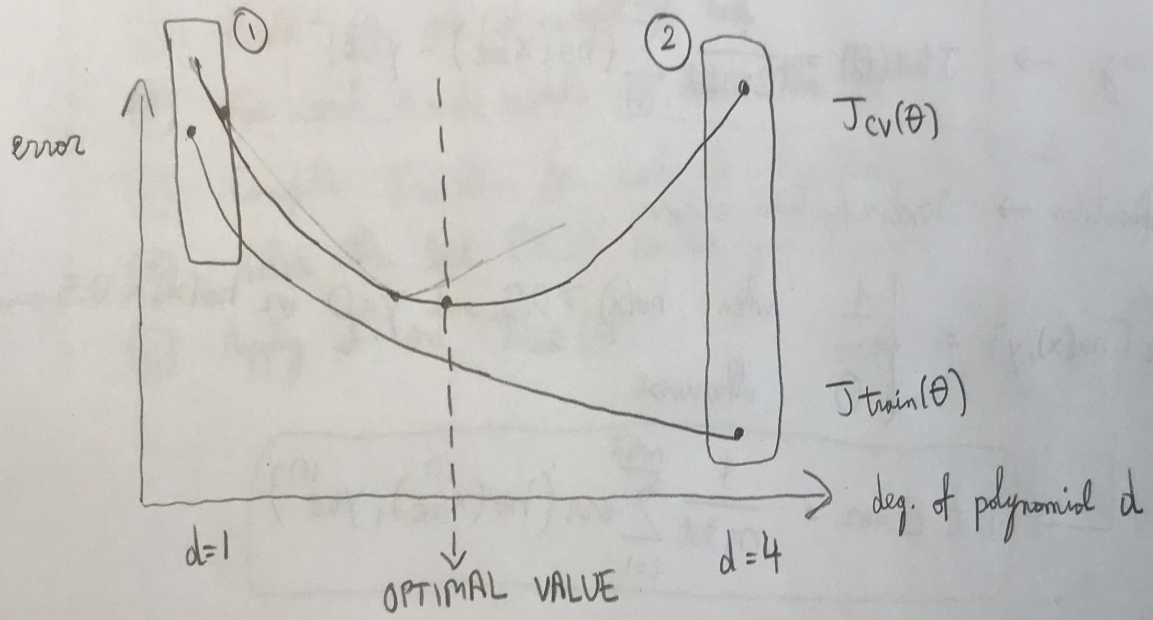
(1) (2)

PICK BEST ONE

Use test set for final evaluation: $J_{test}(\theta^{(3)})$ (3)

BIAS vs VARIANCE:

→ $J_{CV}(\theta)$ or $J_{test}(\theta)$ is high



① HIGH BIAS → underfitting

- → $J_{train}(\theta)$ will be high
- → $J_{CV}(\theta) \approx J_{train}(\theta)$

② HIGH VARIANCE → overfitting

- → $J_{train}(\theta)$ will be low
- → $J_{CV}(\theta) >> J_{train}(\theta)$

(21)

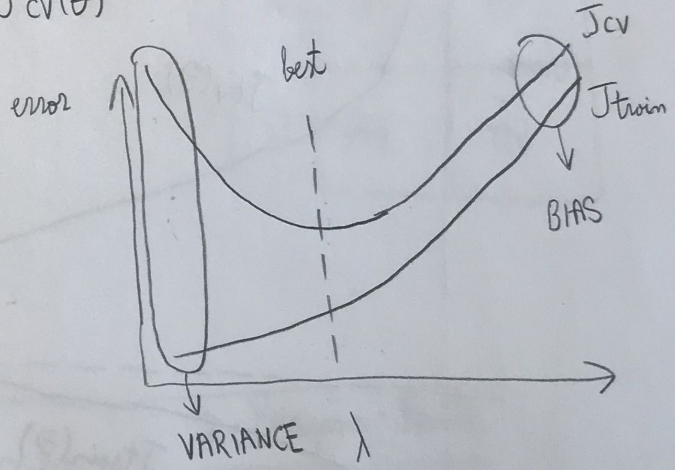
→ Selecting regularization param. λ:

LIN. REG. → $J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \boxed{\frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2}$

- → LARGE λ → high bias → underfitting → we penalize too much
- → SMALL λ → high variance → overfitting → λ is not affecting the optimization

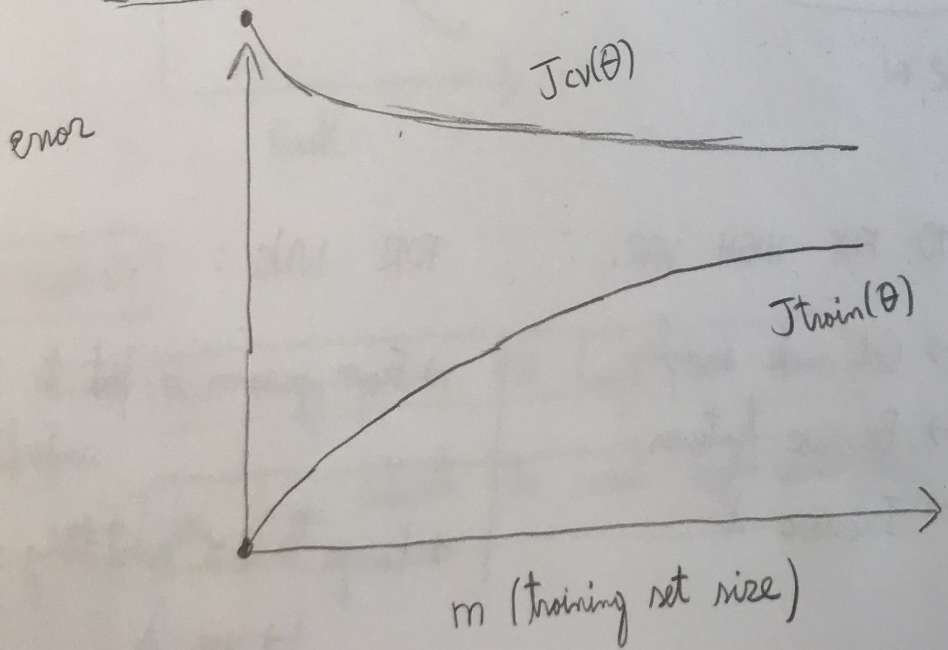
HOW TO CHOOSE λ:

1. Create list of λ values $\lambda = 0, 0.01, 0.02, 0.04, \dots, 10$
2. Create your set of models to test
3. For each λ and models → learn the θ
4. Compute $J_{cv}(\theta)$ for each θ (without reg. term)
5. Select the best (θ, λ) combo on $J_{cv}(\theta)$
6. Apply it on $J_{test}(\theta)$

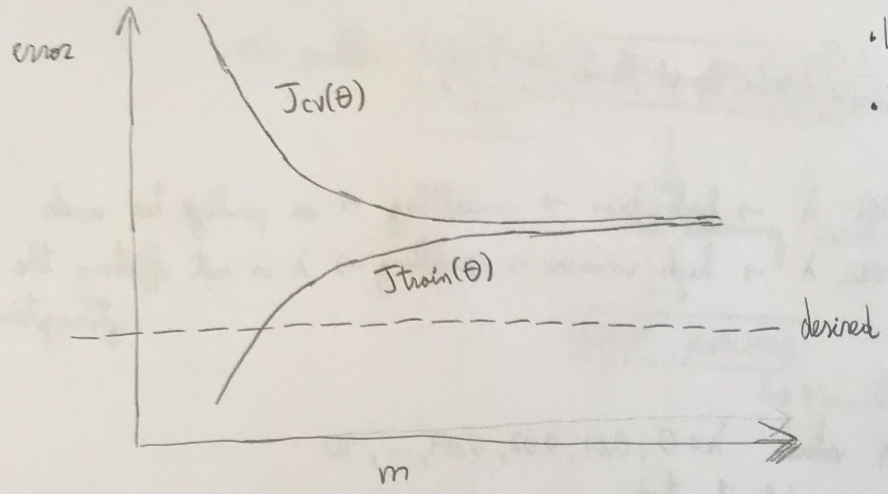


TYPICAL LEARNING CURVE EXPECTED:

$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$



HIGH BIAS:

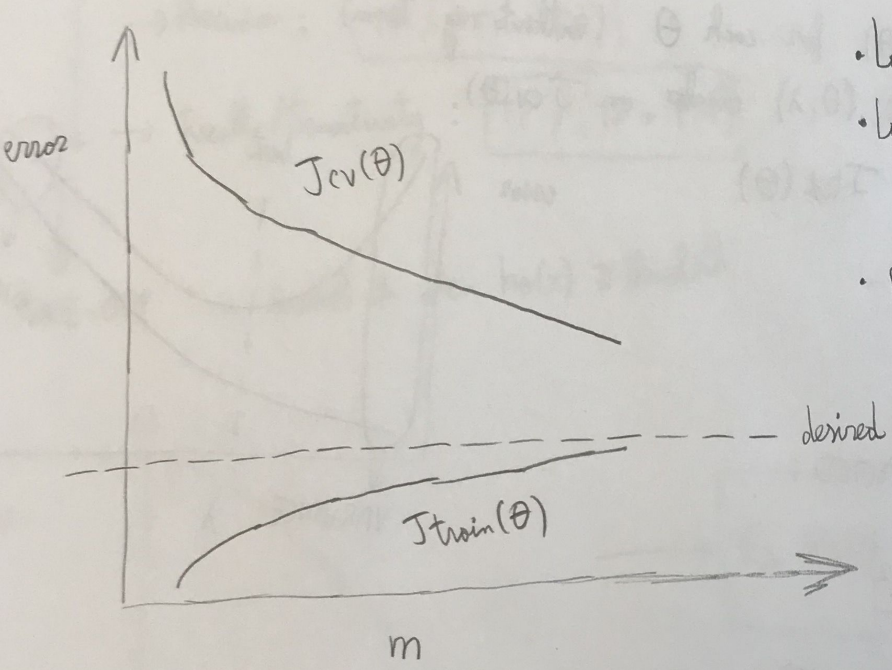


- Low m → J_{train} low and J_{cv} high
- Large m → Both J_{train} and J_{cv} high and $J_{train} \approx J_{cv}$

↓

Having larger dataset WILL NOT HELP

HIGH VARIANCE:



- Low m → J_{train} low and J_{cv} high
- Large m → J_{train} gets higher while J_{cv} get lower and $J_{train} < J_{cv}$
- BUT $|J_{cv} - J_{train}|$ remains high

↓

Having larger dataset WILL PROBABLY HELP

TIPS

FIX HIGH BIAS

→ Add features
→ Add polynomial features
→ Decrease λ

TO FIX HIGH VAR.

→ Get more samples
→ Decrease features
→ Increase λ

FOR NNs:

→ Fewer params. = fast to train but underfitting

→ Larger NN = overfitting and slow
↳ use λ

(23)

ERROR ANALYSIS:

ML problem → Quick simple algo → Test on CV data

↓

Plot learning curves to decide next step

↓

ERROR ANALYSIS

↳ Manually examine the errors on CV examples

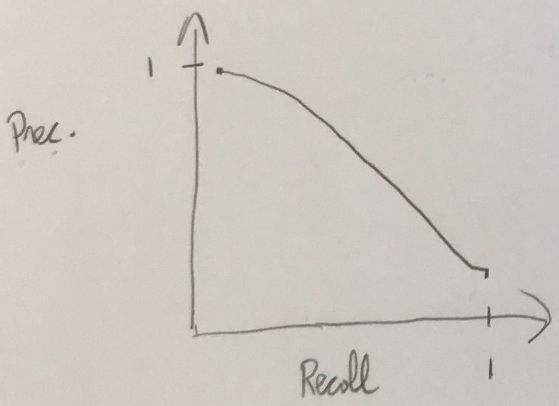
CLASS IMBALANCE (SKEWED CLASSES):

- → Precision: $TP / (TP + FP)$
- → Recall / Sensitivity: $TP / (TP + FN)$

↓

TRADE-OFF → Predict 1 when $h_\theta(x) \geq$ threshold

Predicted \ Ground Truth	1	0
1	TP	FP
0	FN	TN



→ F_1 SCORE: $\frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$

↳ Select this on the CV set!

STRATEGY

↳ Use many params. (to have low bias)

↳ Use huge dataset (to avoid high variance).