Stanford ML

WEEK 6:

DEBUGGING

Suppose you have implemented regularized linear regression to predict housing prices.

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{m}\theta_j^2\right]$$

However, when you test your hypothesis on a new set of houses, you find that it makes unacceptably large errors in its predictions. What should you try next?

Things to try:

- More training examples
- Try smaller set of features
- Try getting additional features
- Adding polynomial features
- Changing λ

} How to decide what to do?

ML DIAGNOSTIC: A test that you can run to identify what is/isn't working with a learning algorithm, and gain guidance on how best to improve its performance

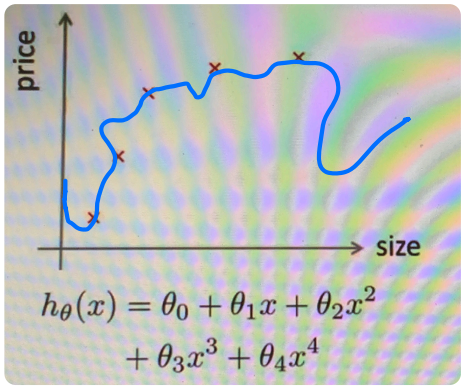
↳ MAY TAKE A LONG TIME BUT WORTH IT!

DIAGNOSTIC 1: Evaluating your Hypothesis

- Low training error may not be a good indication due to OVERFITTING => FAILS TO GENERALIZE

SOLN: For a low # of features we could simply plot $h(\theta)$

⟶ NOT FEASIBLE W/ LARGER DATASETS



ALTERNATIVE:

- Split dataset into a TRAINING and TEST set split ~ 70:30 %. (m, m_{test})
- If there is an order within the set, RANDOMIZE test and training set

Dataset:

Size	Price	
2104	400	Training set
1600	330	
2400	369	
1416	232	
3000	540	
1985	300	
1534	315	
1427	199	Test Set
1380	212	
1494	243	

PROCEDURE: LIN REG.

1.) Learn parameters θ from training data (by minimizing $J(\theta)$)

2.) Compute test set error

$$J_{test}(\theta) = \frac{1}{2m_t} \sum_{i=1}^{m_t} \left\{ h_\theta(x_{test}^{(i)}) - y^{(i)} \right\}^2$$

LIN. REG.

PROCEDURE: LOG. REG.

1.) Learn parameters θ from training data (by minimizing $J(\theta)$)

2.) Compute test set error

$$J_t(\theta) = -\frac{1}{m} \sum_{i=1}^{m_t} \left\{ y_t^{(i)} \log\left(h_\theta(x_t^{(i)})\right) + (1 - y_t^{(i)}) \log\left(1 - h_\theta(x_t^{(i)})\right) \right\}$$

OR we can use the MISCLASSIFICATION ERROR

$$\text{err}(h_\theta(x), y) = \begin{cases} 1 & \text{if } h_\theta(x) \geq 0.5,\ y = 0 \\ & \text{or if } h_\theta(x) \leq 0.5,\ y = 1 \\ & \uparrow \text{MISCLASSIFIED} \\ 0 & \text{otherwise} \end{cases}$$

$$\Rightarrow \boxed{\text{Test error} = \frac{1}{m_t} \sum_{i=1}^{m_t} \text{err}\left\{ h_\theta(x_t^{(i)}), y^{(i)} \right\}}$$

Model Selection

1. $h_\theta(x) = \theta_0 + \theta_1 x$ $d = 1$
2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$ $d = 2$ etc
3. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_3 x^3$

⋮

10. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_{10} x^{10}$

- Almost as if there is another parameter to pick:

$$\boxed{d = \text{degree of polynomial}}$$

- We could find the parameters using each model generating $\theta^{(d=1)}, \theta^{(d=2)}, \ldots, \theta^{(d=10)}$
- Calculate $J_t(\theta)$ for $\forall\ d = 1, \ldots, 10$

$\Rightarrow$ select model w/ lowest J_t e.g. $d = 5$

PROBLEM: Even though $J_t(\theta^{(5)})$ is the smallest, this still MAY NOT generalize well and will likely be an OPTIMISTIC estimate of generalization error. i.e. OUR EXTRA (d) PARAMETER WAS FITTED USING THE TEST SET

$\Rightarrow$ BIASED IN FAVOUR OF TEST SET!

ALTERNATIVE: split the data in 3!

Dataset:

Size	Price
2104	400
1600	330
2400	369
1416	232
3000	540
1985	300
1534	315
1427	199
1380	212
1494	243

TRAINING SET ~ 60%

CROSS VALIDATION SET (CV) m_{cv} 20%

TEST SET ~ 20%

- Instead of using the test set to pick the model we use the CV set

Training error:

$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Cross Validation error:

$$J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$

Test error:

$$J_{test}(\theta) = \frac{1}{2m_{test}}\sum_{i=1}^{m_{test}}(h_\theta(x_{test}^{(i)}) - y_{test}^{(i)})^2$$

1.) Obtain $\theta^{(1)}, \theta^{(2)}, \ldots$ $\theta^{(9)}, \theta^{(10)}$ via $\min_\theta J(\theta)$

2.) Compute $J_{CV}(\theta^{(i)})$ for $\forall\ i = 1, \ldots, d$

3.) Pick lowest $J_{CV}(\theta)$ i.e. $\theta^{(4)}$; $\theta_0 + \theta_1 x_1 + \ldots \theta_4 x^4$

4.) Estimate generalization error for test set $J_t(\theta^{(4)})$

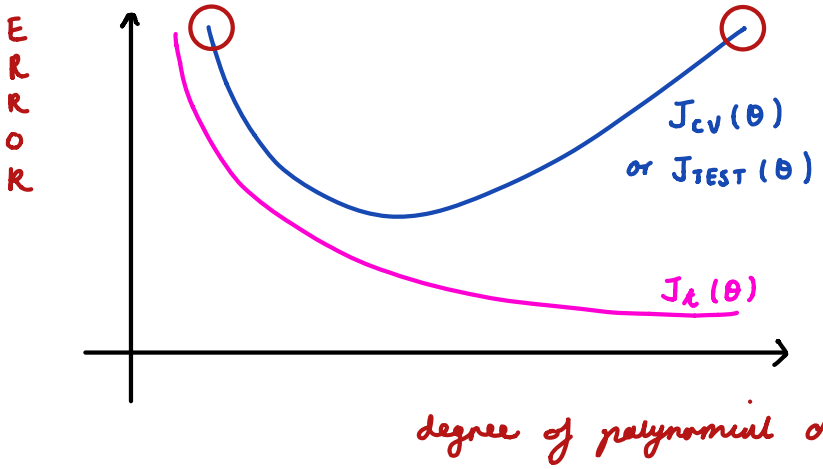
Bias vs. Variance

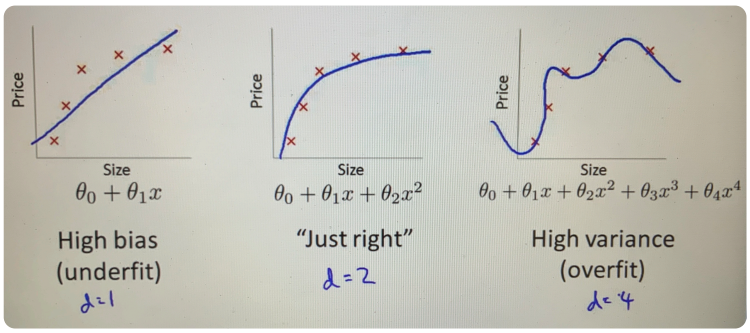
OVER FIT ↙ ↘ UNDER FIT

- Problems usually arise due to one or both aspects in play

TRAINING ERROR: $J_t(\theta) = \frac{1}{2m} \sum_{i=1}^{m} [h_\theta(x^{(i)}) - y^{(i)}]^2$

CROSS-VALIDATION ERROR: $J_{CV}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} [h_\theta(x_{CV}^{(i)}) - y_{CV}^{(i)}]^2$





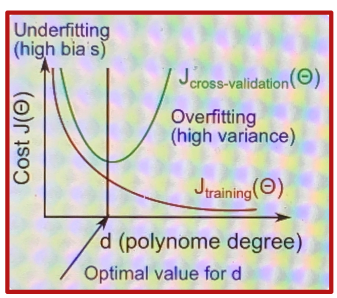
- Large $J_{CV}(\theta)$ for $d = 1$ and $d = 10$

(UNDER) **HIGH BIAS!** ↗ ↖ (OVER) **HIGH VARIANCE!**

⇓

- BOTH J_{CV} and J_{TRAIN} ARE BOTH HIGH

$J_{CV}(\theta) \sim J_{TRAIN}(\theta)$



⇓

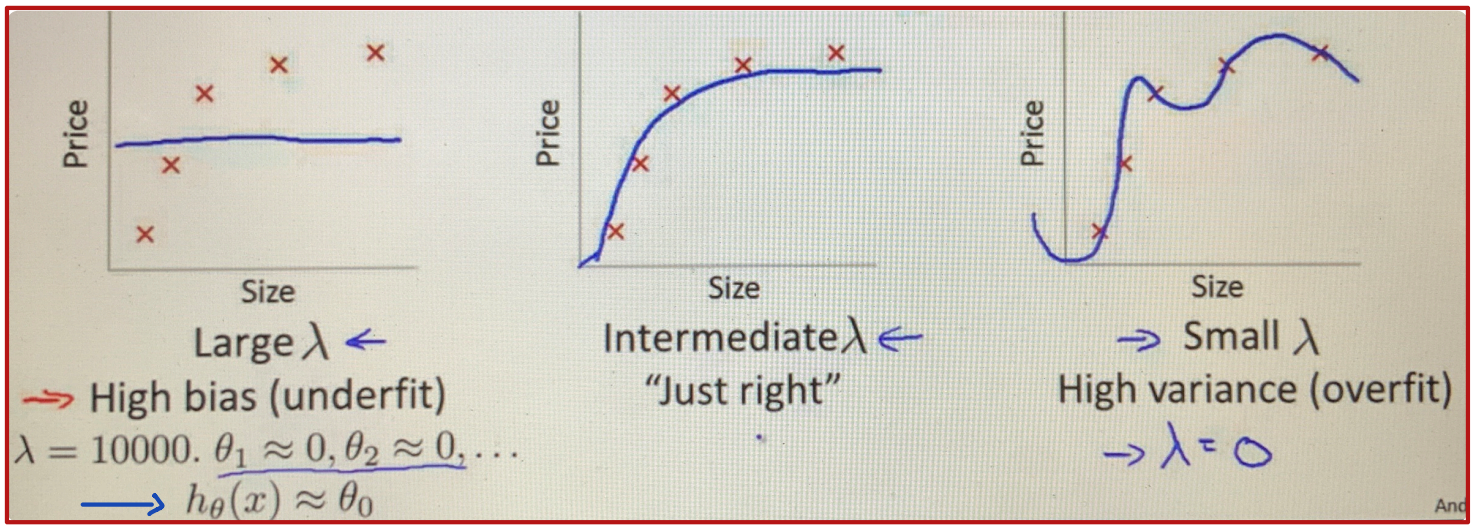
- J_{TRAIN} will be LOW but J_{CV} HIGH

$J_{CV}(\theta) >> J_{TRAIN}(\theta)$

Regularization: BIAS / VARIANCE

- **MODEL:** $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left[h_\theta(x^{(i)}) - y^{(i)}\right]^2 + \boxed{\frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2}$$



Choosing the right λ:

use

* $J_t(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left[h_\theta(x_t^{(i)}) - y_t^{(i)}\right]^2$

Same thing!

* $J_{CV}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left[h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)}\right]^2$

* $J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left[h_\theta(x^{(i)}) - y^{(i)}\right]^2$

* $J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left[h_\theta(x^{(i)}) - y^{(i)}\right]^2 + \boxed{\frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2}$

PROCEDURE:

1.) Consider range of λ values (increase in MULTIPLES of 2)

2.) $\min_\theta J(\theta) \longrightarrow \theta$ for each λ value

3.) Plug in each θ into $J_{CV}(\theta)$

1. Try $\lambda = 0$ ←
2. Try $\lambda = 0.01$
3. Try $\lambda = 0.02$
4. Try $\lambda = 0.04$
5. Try $\lambda = 0.08$

⋮

12. Try $\lambda = 10$

$10 \cdot 24$

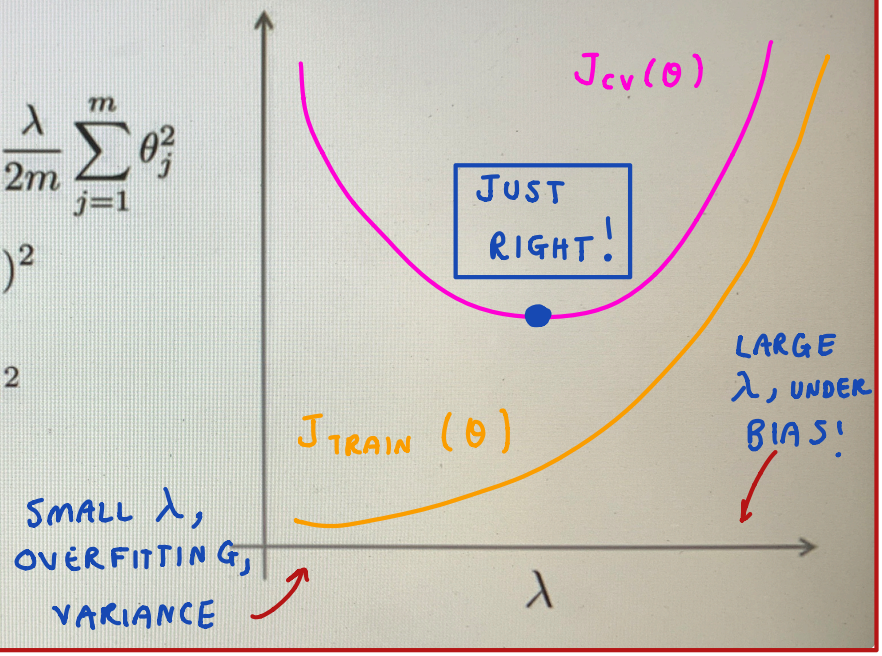
4.) Pick smallest $J_{CV}(\theta)$ i.e $J_{CV}(\theta^{(5)})$

5.) Test error : $J_{test}(\theta^{(5)})$

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

$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$



VARIANCE

- small λ, OVERFITTING
- $J_{CV}(\theta) >> J_{TRAIN}(\theta)$

BIAS

- large λ, UNDERFITTING
- $J_{CV}(\theta) \approx J_{TRAIN}(\theta)$

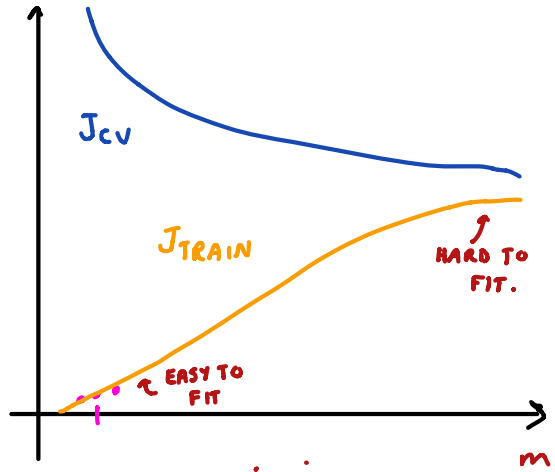
GENERAL ALGORITHM

1. Create a list of lambdas (i.e. λ∈{0,0.01,0.02,0.04,0.08,0.16,0.32,0.64,1.28,2.56,5.12,10.24});
2. Create a set of models with different degrees or any other variants.
3. Iterate through the λs and for each λ go through all the models to learn some Θ.
4. Compute the cross validation error using the learned Θ (computed with λ) on the $J_{CV}(\Theta)$ **without** regularization or λ = 0.
5. Select the best combo that produces the lowest error on the cross validation set.
6. Using the best combo Θ and λ, apply it on $J_{test}(\Theta)$ to see if it has a good generalization of the problem.

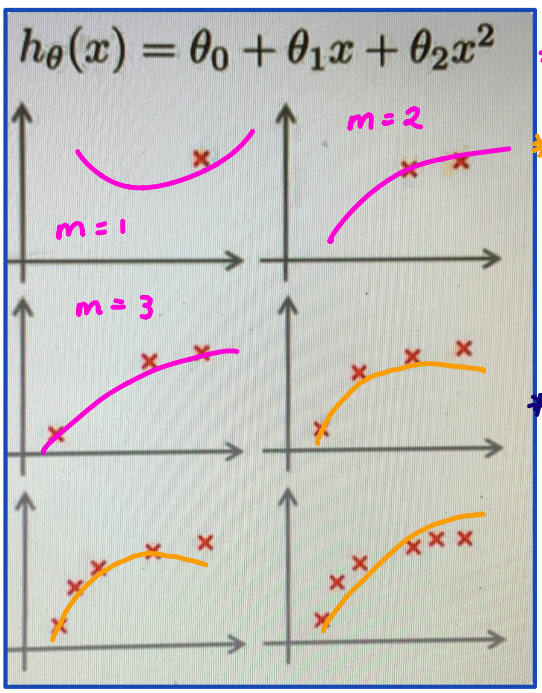
Learning Curves

PLOT:

* $J_{TRAIN}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \{ h_\theta(x^{(i)}) - y^{(i)} \}^2$
* $J_{CV}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \{ h_\theta(x_{CV}^{(i)}) - y_{CV}^{(i)} \}^2$



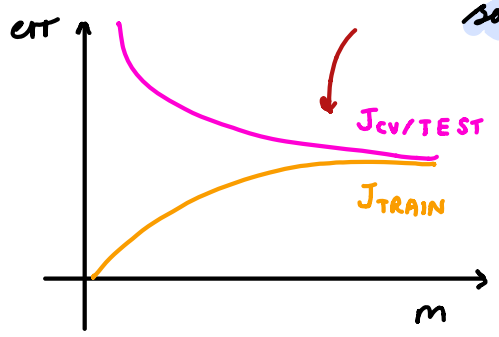
- Artificially reduce the training set size and plot above EQNs as $f(m)$.



* $J_{TRAIN} \approx 0$ w/ REG for low m
* As m get larger, it becomes harder to fit a quadratic function to the data
* Whereas for $J_{CV}(\theta)$, this shows the opposite trend ∴

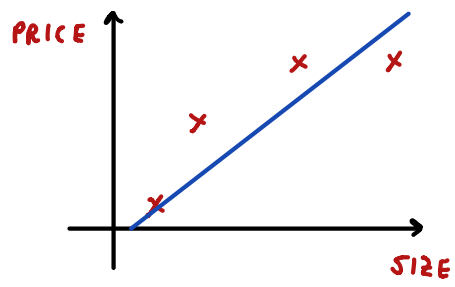
MORE TRAINING EX.	=>	BETTER GENERALIZATION

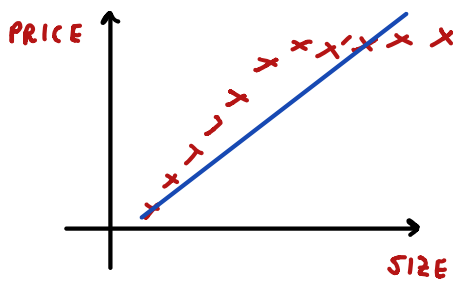
HIGH BIAS



eventually reaches saturation

$h_\theta = \theta_0 + \theta_1 x$

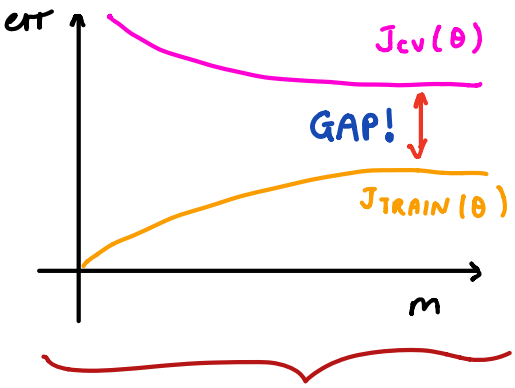




- High errors in general
- Getting more TRAINING DATA WONT HELP FOR HIGH BIAS

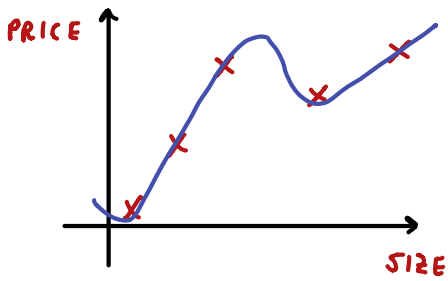
- Even though m has increased, by insisting on a linear hypothesis we get a similar $h_\theta(x)$. } $J_{CV} \approx J_{TRAIN}$

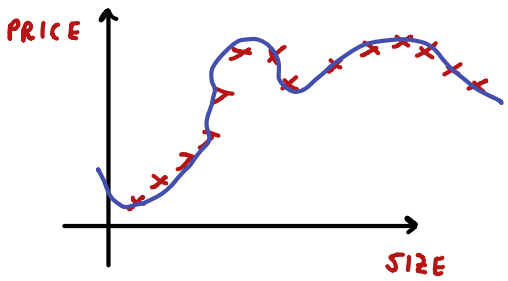
HIGH VARIANCE



(small λ)

$$h_\theta(x) = \theta_0 + \theta_1 x + \dots \theta_{100} x^{100}$$





- Since we are overfitting on the training set, $J_{CV}(\theta)$ will remain large!
- Slightly harder to overfit as m increases ∴ increasing J_{TRAIN}

} $J_{CV} >> J_{TRAIN}$

- As m increases, $J_{CV}(\theta)$ will KEEP DECREASING

∴ IF OVERFITTING AND J_{CV} IS HIGH, GET MORE DATA

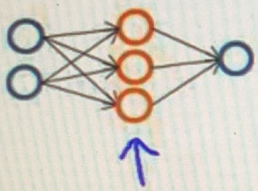
SUMMARY:

Debugging a learning algorithm:
Suppose you have implemented regularized linear regression to predict housing prices. However, when you test your hypothesis in a new set of houses, you find that it makes unacceptably large errors in its prediction. What should you try next?

- Get more training examples → fixes high variance
- Try smaller sets of features → fixes high variance
- Try getting additional features → fixes high bias
- Try adding polynomial features $(x_1^2, x_2^2, x_1x_2, \text{etc})$ → fixes high bias.
- Try decreasing λ → fixes high bias
- Try increasing λ → fixes high variance

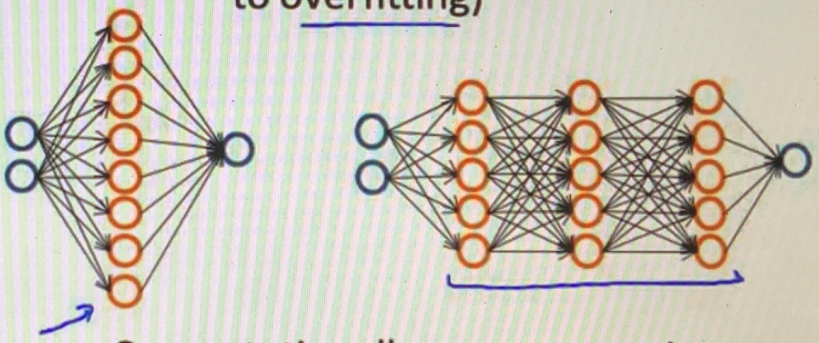
Neural networks and overfitting

"Small" neural network (fewer parameters; more prone to underfitting)



Computationally cheaper

"Large" neural network (more parameters; more prone to overfitting)

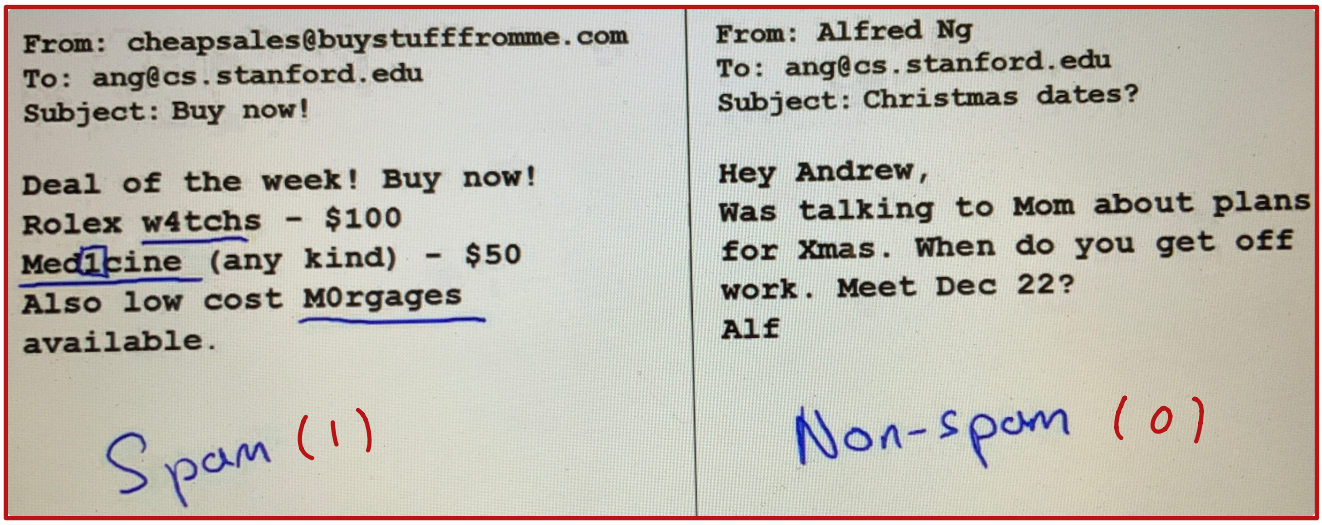


Computationally more expensive.

Use regularization (λ) to address overfitting.

Machine Learning Design

EXAMPLE: Spam Classifier



SUPERVISED LEARNING

- Let there be a labelled training set with spam $\longrightarrow$ 1 and non-spam $\longrightarrow$ 0.
- x = features of e-mail
  - e.g. deal, buy, discount, Cannon
  - Take list of ~ 100 words and check whether the word appears in the email
  - 1 = PRESENT, 0 = NOT-PRESENT

} FEATURE VECTOR $\in \mathbb{R}^{100}$

$$x = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^{100} \; ; \; x_j = \begin{cases} 1 & \text{if word } j \text{ present} \\ 0 & \text{otherwise} \end{cases}$$

NOTE: In practice, take most frequently used n words (10,000 to 50,000) in training set, rather than manually pick 100 words

How to spend your time to minimize error?

1. Collect lots of data e.g. "honeypot" project
2. Develop sophisticated features based on e-mail ROUTING information
3. Develop sophisticated features for message body
4. Develop algorithm to spot mis-spellings

Error Analysis

Recommended Approach:

1. Start with a simple algorithm that you can implement quickly
2. Plot learning curves to decide if more data, more features etc. will help.
3. ERROR ANALYSIS: Manually examine the examples (in CV set) that your algorithm made errors on. See if SYSTEMATIC TREND can be spotted on errors.

m_{CV} = 500 examples in cross validation set
Algorithm misclassifies 100 emails.
Manually examine the 100 errors, and categorize them based on:
→ (i) What type of email it is — pharma, replica, steal passwords, ...
→ (ii) What cues (features) you think would have helped the algorithm classify them correctly.

Pharma: 12
Replica/fake: 4
Steal passwords: 53
Other: 31

WORTH FOCUSSING ON

→ Deliberate misspellings: 5
(m0rgage, med1cine, etc.)
→ Unusual email routing: 16
→ Unusual (spamming) punctuation: 32

- Always have a numerical evaluation of your algorithm e.g.

Should discount/discounts/discounted/discounting be treated as the same word?
Can use "stemming" software (E.g. "Porter stemmer")
universe/university.
Error analysis may not be helpful for deciding if this is likely to improve performance. Only solution is to try it and see if it works.
Need numerical evaluation (e.g., cross validation error) of algorithm's performance with and without stemming.
Without stemming: 5% error With stemming: 3% error

Skewed Classes

Ex:

* Train logistic regression model $h_\theta(x)$ ($y = 1$ if cancer, $y = 0$ otherwise)
* Find 1% error on test set
* ONLY 0.50% HAVE CANCER

↳ SKEWED CLASSES

Far greater no. of patients without cancer than with it.

- In the skewed classes regime it is not always clear whether an improvement in accuracy was due to a beneficial change to the algorithm, or by mathematical fluke due to the skewed class

ALTERNATIVE: Precision / Recall

* $y = 1$ in presence of rare class that we want to detect

ACTUAL CLASS

PREDICTED CLASS	1	0
1	TRUE POSITIVE	FALSE POSITIVE
0	FALSE NEGATIVE	TRUE NEGATIVE

PRECISION

- Of all patients we predicted $y=1$, what fraction actually have cancer?

$$p = \frac{\text{True Positives}}{\text{True pos} + \text{False pos.}}$$

RECALL

- Of all patients that actually have cancer, how many did we predict correctly?

$$r = \frac{\text{True Positives}}{\text{True pos} + \text{False neg.}}$$

$$\text{Precision} = \frac{\text{True positives}}{\text{\# predicted as positive}} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}}$$

$$\text{Recall} = \frac{\text{True positives}}{\text{\# actual positives}} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}}$$

- Having an algorithm with both high recall and high precision confirm we have a classifier that will generalize well to new data.

→ FIXES SKEWED CLASSES