Subject: Advice For applying ML / /

Evaluate a model:

- Split data into parts

size	price

70% training set: $(x^1_{train}, y^1_{train})$ ... $(x^{m_{train}}, y^{m_{train}})$

30% test set: (x^1_{test}, y^2_{test}) ... $(x^{m_{test}}, y^{m_{test}})$

Linear Regression:

Minimize Cost:

$$J(\vec{w}, b) = \left[\frac{1}{2m_{train}} \sum_{i=1}^{m_{train}} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2m_{train}} \sum_{j=1}^{n} w_j^2 \right]$$

Compute test error:

$$J_{test}(\vec{w}, b) = \frac{1}{2m_{test}} \left[\sum_{j=1}^{m_{test}} \left(f_{\vec{w},b}(\vec{x}^{(i)}_{test}) - y^{(i)}_{test} \right)^2 \right]$$

Compute training error:

$$J_{train}(\vec{w}, b) = \frac{1}{2m_{train}} \left[\sum_{j=1}^{m_{train}} \left(f_{\vec{w},b}(\vec{x}^{(i)}_{train}) - y^{(i)}_{train} \right)^2 \right]$$

Model Selection:

d=1 1) $f_{\vec{w},b} = w_1 x + b \longrightarrow J_{test}(w^{<1>}, b^{<1>})$

d=2 2) $f_{\vec{w},b} = w_1 x + w_2 x^2 + b \longrightarrow J_{test}(w^{<2>}, b^{<2>})$

⋮

d=10 10) $f_{\vec{w},b} = w_1 x + w_2 x^2 + \dots + w_{10} x^{10} + b \longrightarrow J_{test}(w^{<10>}, b^{<10>})$

↳ since, we are also using d as a parameter, then this d might overfit the test set

Subject: / /

- Choose one with lowest J_{test}
  - ↳ This is NOT good as the model will overfit the test set and will NOT generalize truly

Training / cross validation / test set

↳ validation / development / dev

- Split data into parts

size	price

60% training set: $(x^1_{train}, y^1_{train})$ ⋮ $(x^{m_{train}}, y^{m_{train}})$

20% cross validation: (x^1_{cv}, y^1_{cv}) ⋮ $(x^{m_{cv}}, y^{m_{cv}})$

test set 20%: (x^1_{test}, y^1_{test}) ⋮ $(x^{m_{test}}, y^{m_{test}})$

Compute test error:

$$J_{test}(\vec{w}, b) = \frac{1}{2m_{test}} \left[\sum_{j=1}^{m_{test}} \left(f_{\vec{w},b}(\vec{x}^{(i)}_{test}) - y^{(i)}_{test} \right)^2 \right]$$

Compute training error:

$$J_{train}(\vec{w}, b) = \frac{1}{2m_{train}} \left[\sum_{j=1}^{m_{train}} \left(f_{\vec{w},b}(\vec{x}^{(i)}_{train}) - y^{(i)}_{train} \right)^2 \right]$$

Compute CV error:

$$J_{cv}(\vec{w}, b) = \frac{1}{2m_{cv}} \left[\sum_{j=1}^{m_{cv}} \left(f_{\vec{w},b}(\vec{x}^{(i)}_{cv}) - y^{(i)}_{cv} \right)^2 \right]$$

Subject: / /

Model Selection:

1) $f_{\vec{w},b} = w_1x + b \longrightarrow J_{cv}(w^{<1>}, b^{<1>})$
2) $f_{\vec{w},b} = w_1x + w_2x^2 + b \longrightarrow J_{cv}(w^{<2>}, b^{<2>})$

$\vdots$

10) $f_{\vec{w},b} = w_1x + w_2x^2 + \dots + w_{10}x^{10} + b \longrightarrow J_{cv}(w^{<10>}, b^{<10>})$

- Choose one with lowest J_{cv}.
- Estimate generalization error using J_{test}

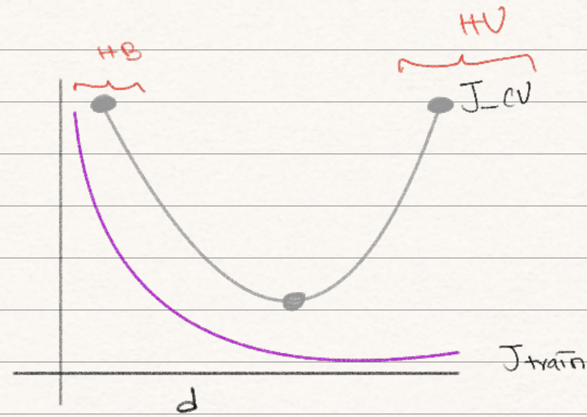
Diagnosing Variance and bias:

High bias (Under fit)	"Just right"	High Variance (over fit)
J_train is high	J_train is low	J_train is low
J_cv is high	J_cv is low	J_cv is high
$J_{train} \approx J\_cv$		$J_{cv} >> J_{train}$

High variance & bias

J_{train} is high

$J_{cv} >> J_{train}$



Subject:

Regularization and bias/Variance:

Model: $f_{\vec{w},b}(x) = w_1x + w_2x^2 + w_3x^3 + w_4x^4 + b$

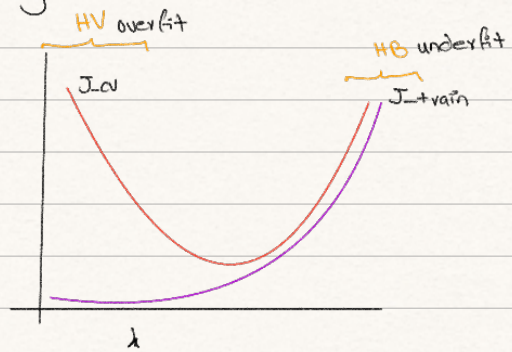
$$J(\vec{w},b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(x^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{m} w_j^2$$

Large λ	small λ	Intermediate λ
high bias (under fit)	High variance (over fit)	

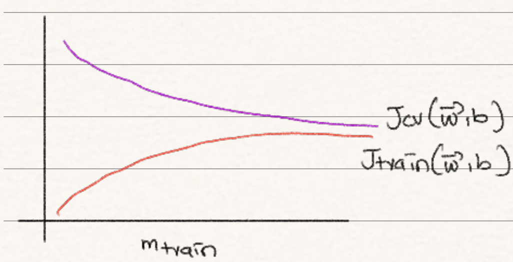
1. Try $\lambda = 0 \longrightarrow \min_{\vec{w},b} J(\vec{w},b) \longrightarrow w^{<1>}, b^{<1>} \longrightarrow J_{cv}(w^{<1>}, b^{<1>})$

$\vdots$

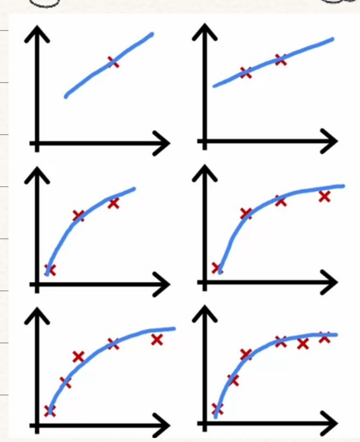
12. Try $\lambda \approx 10 \longrightarrow \min_{\vec{w},b} J(\vec{w},b) \longrightarrow w^{<12>}, b^{<12>} \longrightarrow J_{cv}(w^{<12>}, b^{<12>})$



Learning Curves:

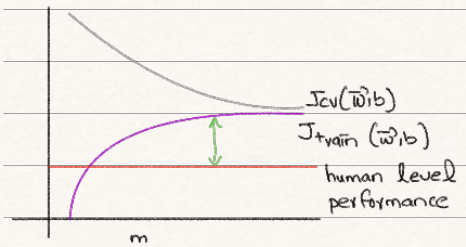


Why Jtrain gets bigger?

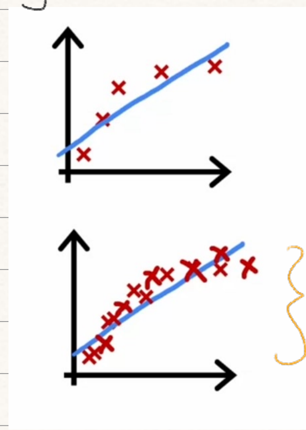


Subject:

High bias:

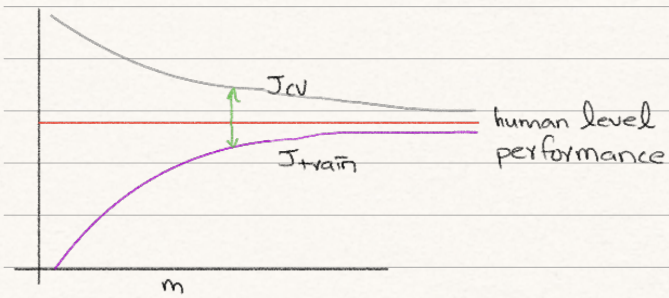


Why curves flatten?



regardless of amount of data, a linear model is too simple & won't change much

High Variance:



- More data likely to help

What to do next?

High Bias:

- Try using more features
- Try adding polynomial features
- Decrease λ

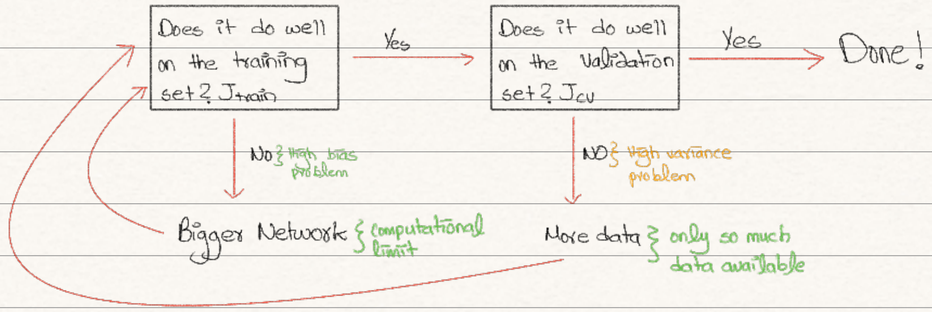
High Variance:

- Get more data
- Try using less features
- Increase λ

Subject: / /

Bias & variance in nueral network:

Large networks trained on small/moderate data are low bias machines



- A larger Nueral Network which is regularized correctly will perform just as good or better than smaller. So, it never hurts to go bigger!

  slows down the algo ☹

Regularized:

λ

layer_1 = Dense (25, activation ="relu", kernel_regularizer = L2 (0.01))

Machine learning development process:

Error analysis:

$m_{cv} = 500$

Algo misclassifies 100 of them

Manually examine the 100 examples and group them based on common traits.

if misclassified amount is very large, try random sampling

Data Augmentation:

- Modifying existing training example to create a

Subject: / /

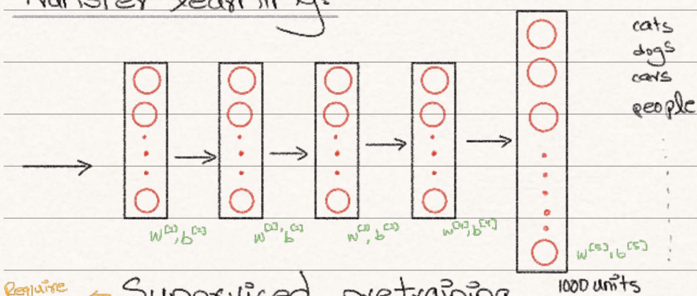
new training examples.

- Distortion introduced should be represented of the type of noise/distortions in the test set. { Distortions in test set & must be meaningful

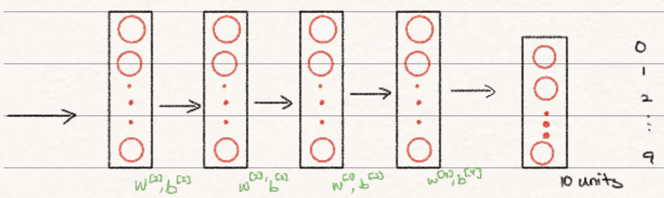
Data synthesis:

- Create new data on your own.

Transfer learning:



Require large data ← Supervised pretraining
1 million samples
↳ fairly available on internet



fine tuning

You want to train model to recognize digits. But You don't have data. Train model on some other data like recognize dogs, cats etc. Now, copy the nueral network & use all the previous parametres but train last layer.

Option 1: only train output layers parametres.
Option 2: Train all parametres. Initialize with above nn parametres.

Why does this work?

layer 1 : detect edges
layer 2: detect corners
layer 3: detect curves/basic shapes } learn generic features

- Use the same input type.

Subject: / /

Skewed datasets:

- Ratio of + & − is NOT 50/50

Precision/Recall:

$y=1 \longrightarrow$ presence of a rare class

Confusion matrix:

True positive

		Actual class 1	Actual class 0
Predicted class	1	15 (True positive)	5 (False positive)
Predicted class	0	10 (False negative)	70 (True negative)

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

Of those we predicted as $y=1$, how many actually has the disease?

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

Of those that have the disease, how many did we correctly predict?

Trading off Precision & Recall:

Raising threshold increases precision but lower recall and vice versa

F1 score:

How to compare precision/Recall numbers?

$$\text{Avg} = \frac{P+R}{2}$$

Subject:

	P	R	Avg	F1 Score
Algo 1	0.5	0.4	0.45	0.444
Algo 2	0.7	0.1	0.4	0.175
Algo 3	0.02	1.0	0.501	0.0392

Better of the three

$$\text{F1 score} = \frac{1}{\frac{1}{2}\left(\frac{1}{P}+\frac{1}{R}\right)} = 2\frac{PR}{P+R}$$

/ /