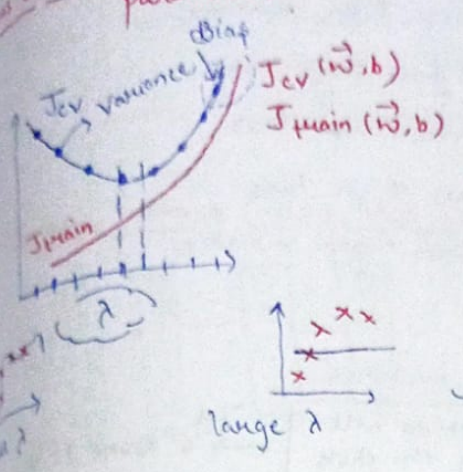
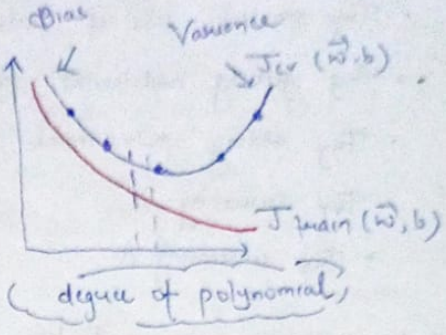
Week_2 Machine learning

Choosing Model

Bias and variance as a function of regularization parameters?





large λ

comparision

* Speech recognition example

this is high bias problem
- Human level performance : 10.6%
- Training error J_{train} : 10.8%

0.2% that's not bad

high variance problem
- Cross validation error J_{cv} : 14.8%

4% it's high

Establishing a baseline level of performance

what is the level of error you can reasonably hope to get to?

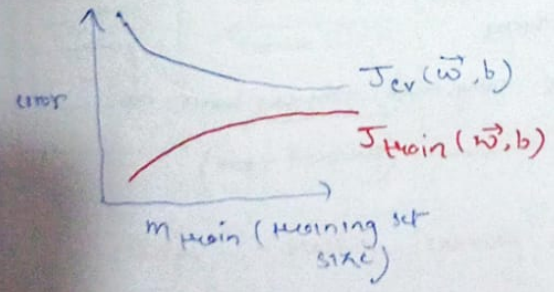
- Human level performance
- Competing algorithms performance
- Guess based on experience.

- Learning Curves.

$f_{\vec{w},b}(x) = w_1x + w_2x^2 + b$

J_{train} = training error

J_{cv} = cross validation error



- Conclusion

If your problem is high Bias Then giving more training data does not shows much effect.

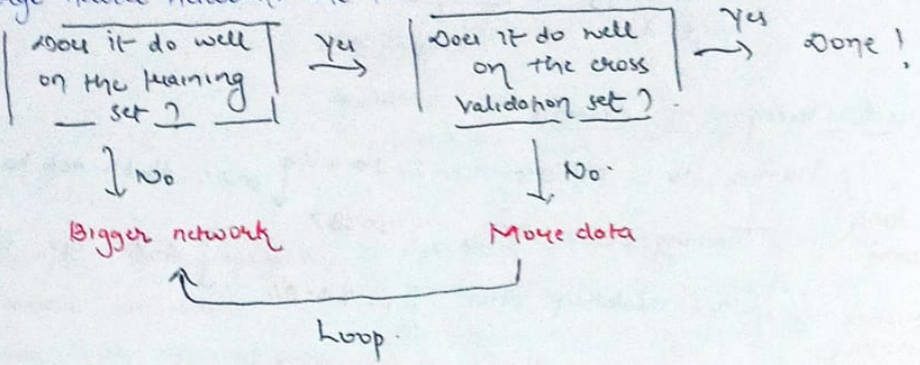
- In case of high variance then increasing dataset size helps a lot.

Q → What do 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 → fixes high bias
- Try decreasing λ → fixes high bias
- Try increasing λ → fixes high variance

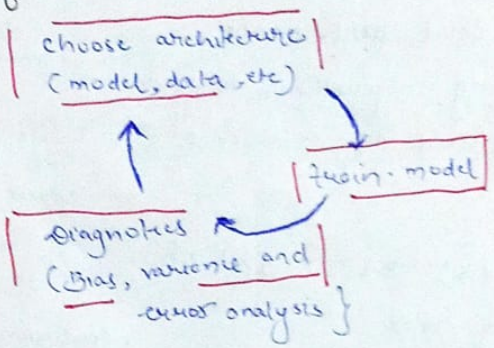
- **Neural networks and bias variance**

Large neural networks are low bias machines



- **ML development process**

⇒ Iterative loop of ML development



- **Error analysis :**

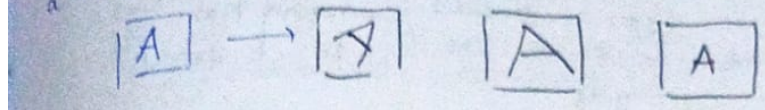
m_{cv} = 500 examples in cross validation set

Algorithm misclassifies 100 of them

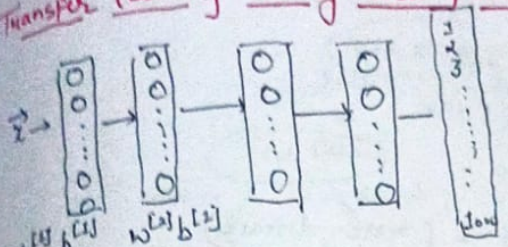
Manually examine 100 ex and categorize them based on common traits.

Data augmentation:

Augmentation: modifying an existing training example to create a new training examples.

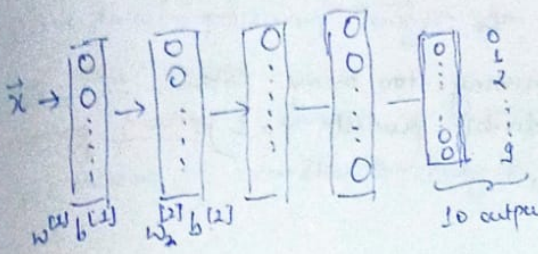


Transfer learning: using data from a different task



suppose you don't have cat data

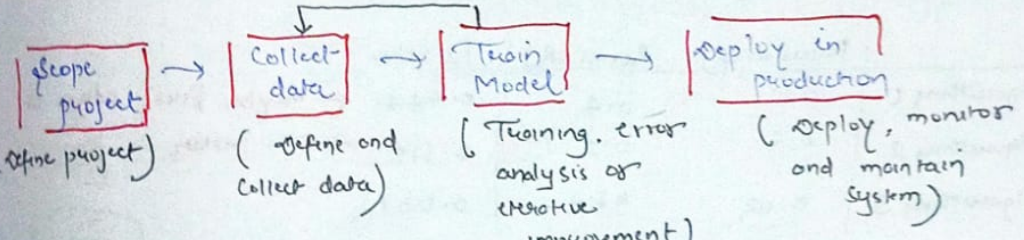
That's why we are training in million random data.



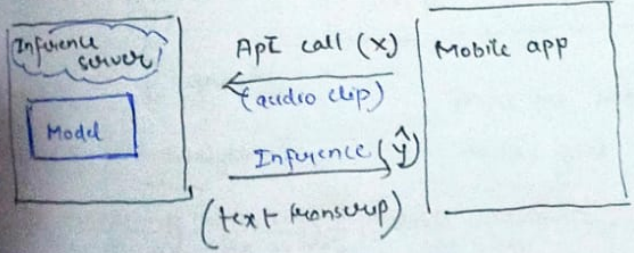
option 1: only train output layer parameters

option 2: train all parameters.

- Full cycle of a machine learning project



- Deployment.



MLOps :- Machine learning operations.

- Precision / Recall

Ex: $y=1$ is presence of rare class we want to detect

predicted class \ actual class	1	0
1	T.P 15	F.P 5
0	F.negative 10	T.N 70

Precision: True positive / predicted positive

$\Rightarrow \frac{\text{True positive}}{\text{True pos + False pos}} = \frac{15}{15+5} = 0.75$

Recall: True positive / actual positive

$\Rightarrow \frac{\text{True positive}}{\text{True pos + False neg}} = \frac{15}{15+10} = 0.6$

§ Suppose we want to predict $y=1$ (rare disease) only if very confident → higher precision, lower recall

Suppose we want to avoid missing too many case of rare disease (when in doubt predict $y=1$) → lower precision, higher recall

F_1 Score:

How to compare precision / recall numbers.

$$F_1 \text{ score} = \frac{1}{\frac{1}{2}\left(\frac{1}{P}+\frac{1}{R}\right)} \Rightarrow 2\frac{PR}{P+R}$$ → Harmonic Mean

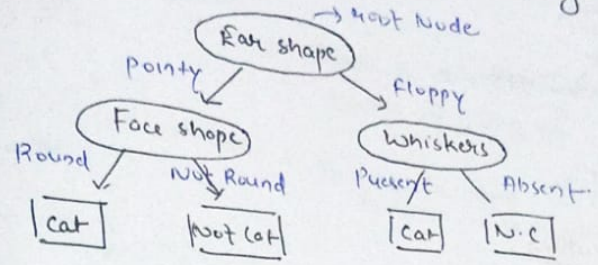
Q

	Precision (P)	Recall (R)	F_1 Score
Algorithm 1	0.5	0.4	0.444 ← maybe first algo is better.
Algorithm 2	0.7	0.1	0.175
Algorithm 3	0.02	1.0	0.0392

Decision Tree Model

eg cat classification example

Ear shape	Face shape	Whiskers	Cat
Pointy	Round	Present	1
Floppy	Not round	Absent	0



Decision Tree learning

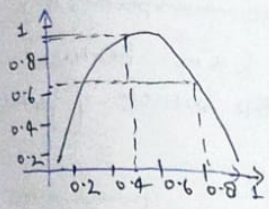
Measuring purity : Entropy as a measure of impurity.

p_1 = fraction of examples that are cat

$p_1 = 3/6 \quad H(p_1) = 1$

$p_1 = 2/6 \quad H(p_1) = 0.92$

$p_1 = 5/6 \quad H(p_1) = 0.65$



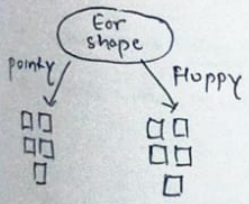
$p_0 = 1 - p_1$

$H(p_1) = -p_1 \log_2(p_1) - p_0 \log_2(p_0)$

$= -p_1 \log_2(p_1) - (1-p_1)\log_2(1-p_1)$

Note "$0 \log(0)$" $= 0$

- **Choosing a split**: Information gain



Information gain

$= H(p_1^{root}) - (w^{left} H(p_1^{left}) + w^{right} H(p_1^{right}))$

$p_1^{left} = 4/5 \quad p_1^{right} = 1/5$

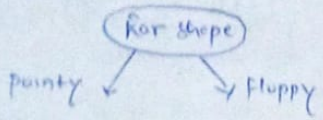
$w^{left} = 5/10 \quad w^{right} = 5/10$

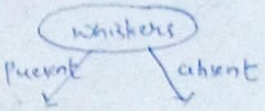
More the reduction in entropy more good it is to split feature.

- **Keep repeating splitting process until stopping criteria is met**:
  - When a node is 100% one class
  - When splitting a node will result in the tree exceeding a maximum depth.
  - Information gain from additional split is less than threshold
  - When number of examples in a node is below a threshold.

- Using one-hot encoding of categorical features.
  If a categorical features can take on k values, create k binary features (0 to 1 valued)

- Tree ensembles.
  Trees are highly sensitive to small changes of the data.





Collection of Trees

- Random Forest Algorithm
  Given training set of size m
  For b = 1 to B:
  Use sampling with replacement to create a new training set of size m. Train a decision tree on the new dataset.

Randomizing the feature choice
At each node, when choosing a feature to use to split. If n features are available, pick a random subset of k < n features and allow the algorithm to only choose from the subset of features.

$$k = \sqrt{n}$$

XGBoost:
We will focus on subset not doing properly. And then create a new decision tree boosting that particular part to perform well. {extreme gradient Boosting}