$$L = -y\log a - (1-y)\log(1-a)$$

$$\frac{dL}{da} = -y\frac{1}{a}(1) - (1-y)\frac{-1}{1-a}$$

$$= \frac{-y}{a} + \frac{1-y}{1-a} = \frac{-y(1-a) + a(1-y)}{a(1-a)}$$

$$\frac{ay - y - ax + a}{a(1-a)}$$

$$= \frac{a-y}{a(1-a)}$$

$$\frac{da}{dz} = \frac{d}{dz}\sigma(z)$$

$$= \sigma(z)(1-\sigma(z))$$

$$\frac{dL}{dz} = \frac{dL}{da}\frac{da}{dz}$$

$$= \frac{(a-y)\cdot a(1-a)}{a(1-a)}$$

$$= a - y$$

$$w = w - \alpha\frac{d}{dw}J(w,b) \rightarrow \frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})x^{(i)}$$

$$b = b - \alpha\frac{d}{db}J(w,b) \rightarrow \frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})$$

$$\frac{\partial}{\partial w}J(w,b) = \frac{d}{dw}\frac{1}{2m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})^2$$

$$= \frac{d}{dw}\frac{1}{2m}\sum_{i=1}^{m}(wx^{(i)} + b - y^{(i)})^2$$

$$= \frac{1}{2m}\sum_{i=1}^{m}(wx^{(i)} + b - y^{(i)})\,2x^{(i)}$$

$$= \frac{1}{m}\sum_{i=1}^{m}(wx^{(i)} + b - y^{(i)})\,x^{(i)}$$

$$\frac{\partial}{\partial b} J(w,b) = \frac{\partial}{\partial b} \frac{1}{2m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})^2$$

$$= \frac{\partial}{\partial b} \frac{1}{2m} \sum_{i=1}^{m} (wx^{(i)} + b - y^{(i)})^2$$

$$= \frac{1}{2m} \sum_{i=1}^{m} (wx^{(i)} + b - y^{(i)}) \not{2}$$

$$= \frac{1}{m} \sum_{i=1}^{m} (wx^{(i)} + b - y^{(i)})$$

$$= \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})$$

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(w_1, w_2, \dots, w_n, b)$$

$$b = b - \alpha \frac{\partial}{\partial b} J(w_1, w_2, \dots, w_n, b)$$

$$\Rightarrow w_j = w_j - \alpha \frac{d}{dw_j} J(\vec{w}, b)$$

$$b = b - \alpha \frac{d}{db} J(\vec{w}, b)$$

$$w = w - \alpha \frac{d}{dw} J(w, b)$$

$$\rightarrow \quad w - \alpha \left(\frac{1}{2m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})^2 \right)$$

$$\Rightarrow w - \alpha \frac{1}{2m} \sum_{i=1}^{m} (w^{(i)} x^{(i)} - y^{(i)}) R(x^{(i)})$$

$$\Rightarrow w - \alpha \frac{1}{m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)}) x^{(i)}$$

$$b = b - \alpha \frac{\partial}{\partial b} J(w,b)$$

$$= b - \alpha \frac{1}{2m} \sum_{i=1}^{m} f_{w,b}(x^{(i)}) - y^{(i)})$$

$$w = w - \alpha \frac{d}{dw} J(w,b)$$

$$= w - \alpha \left(\frac{1}{2m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})^2 \right)$$

$$= w - \alpha \frac{d}{dw}\left(\frac{1}{2m}\sum_{i=1}^{m}(wx^{(i)} - y^{(i)})^2\right)$$

$$= w - \alpha \frac{1}{2m} \times 2 \times \sum_{i=1}^{m}(wx^{(i)} - y^{(i)})x^{(i)}$$

$$b = b - \alpha \frac{1}{2m}\sum_{i=1}^{m}(wx^{(i)} - y^{(i)})$$

- Mean Normalization

$$\frac{x - \mu}{\max - \min}$$

- Z score Normalization (with standard deviation)

$$\frac{x - \mu}{\sigma}$$

- Feature Scaling

$-1 \le x_i \le 1$

$-3 \le x_i \le 3$

$-0.3 \le x_i \le 0.3$

} all acceptable

if different values of features ranging too large.
we have to do feature scaling

$0 \le x_1 \le 3$	okay, no rescaling
$-2 \le x_2 \le 0.5$	okay, no rescaling
$-100 \le x_3 \le 100$	too large → rescale
$-0.001 \le x_4 \le 0.001$	too small → rescale
$98.6 \le x_5 \le 105$	too large → rescale

} to prevent local minimum for gradient descent

- Gradient Descent

$$w_j = w_j - \eta \frac{\partial L}{\partial w_j}, \quad L = J(\vec{w}, b)$$

$$b = b - \eta \frac{\partial L}{\partial b}$$

Feature Engineering

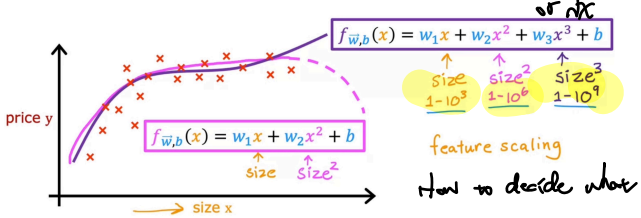
1. Utilizing known features to create new features
   By combining or transforming

Polynomial Regression

→ Multiple Linear Regression + Feature Engineering

goal: fit curves

→ feature scaling is needed

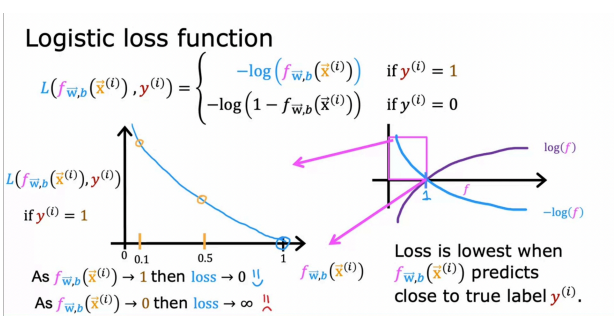


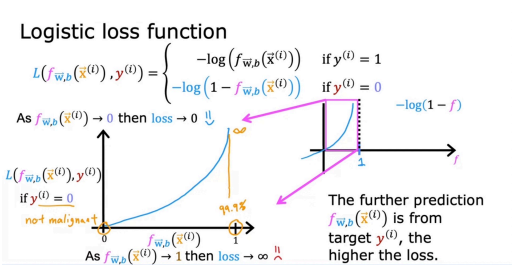
How to decide what features to use? In course 2

Logistic loss function

$$L\left(f_{(\vec{w},b)}(x^{(i)}), y^{(i)}\right) = \begin{cases} -\log\left(f_{\vec{w},b}(x^{(i)})\right) & \text{if } y^{(i)}=1 \\ -\log\left(1-f_{\vec{w},b}(x^{(i)})\right) & \text{if } y^{(i)}=0 \end{cases}$$

$\in [0,1]$





Cost

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

$$= \frac{1}{m} \sum_{i=1}^{m} L\left(f_{w,b}(x^{(i)}), y^{(i)} \right)$$

$\frac{1}{m}$: mean all the loss to obtain the overall cost

$$= \begin{cases} -\log\left(f_{w,b}(x^{(i)})\right) & \text{if } y^{(i)}=1 \\ -\log\left(1 - f_{w,b}(x^{(i)})\right) & \text{if } y^{(i)}=0 \end{cases}$$

plot of the cost func is "convex", so GD is applied to find optimal

* if using the MSE for logistic regression, the loss is non-convex, which cause local minima

$$\rightarrow loss\left(f_{w,b}(x^{(i)}), y^{(i)}\right) = -y^{(i)} \log\left(f_{w,b}(x^{(i)})\right) - \left(1-y^{(i)}\right) \log\left(1 - f_{w,b}(x^{(i)})\right)$$

♡ Address Overfitting

1. More data collection to train model
2. feature selection (throwing some features) — but might lose useful features
3. Regularization: on $w_1, w_2, \dots w_n$ (Reduce the sizes of params)

◎ Regularization on linear regression

1. Penalize all the w_j params (because we don't know which features should be included/excluded)

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

$\lambda \geq 0$ regularization parameter

→ minimize the original cost

$$J(\vec{w}, b) = \min \left[\underbrace{\frac{1}{2m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(x^{(i)}) - y^{(i)} \right)}_{\text{mean square error}} + \underbrace{\frac{\lambda}{2m} \sum_{j=1}^{n} w_j^2}_{\text{regularization term}} \right]$$

regularization term

keep w_j small

if λ is very big, to reduce original error → w_j would be small

Gradient Descent

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

$$b = b - \alpha \frac{\partial}{\partial b} J(\vec{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \left(f_{w,b}(x^{(i)}) - y^{(i)}\right)$$ don't have to regularize "b"

for GD:

$$\text{loop } \{ \; w_j = w_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} \left(f_{w,b}(x^{(i)}) - y^{(i)}\right) x_j^{(i)} + \frac{\lambda}{m} w_j^{(i)} \right]$$

$$b = b - \alpha \left(\frac{1}{m} \sum_{i=1}^{m} \left(f_{w,b}(x^{(i)}) - y^{(i)}\right) \right) \}$$

$$w_j = \underbrace{w_j - \alpha \frac{\lambda}{m} w_j}_{w_j\left(1 - \alpha \frac{\lambda}{m}\right)} - \underbrace{\alpha \frac{1}{m} \sum_{i=1}^{m} \left(f_{w,b}(x^{(i)}) - y^{(i)}\right) x_j^{(i)}}_{\text{usual update}}$$

slightly less than 1

$\alpha \frac{\lambda}{m}$

$0.01 \frac{1}{50} = 0.002 \rightarrow 1 - \alpha \frac{\lambda}{m} = 1 - 0.002 = 0.9998$

Regularization on logistic regression

$$f_{w,b}(x^{(i)}) = \frac{1}{1 + e^{-z}}$$

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

Gradient Descent

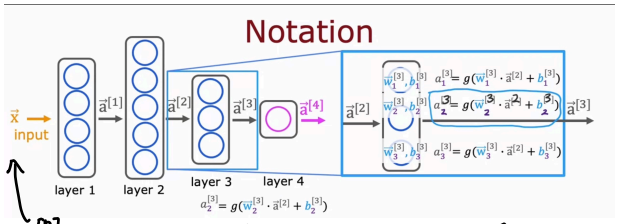
$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w}, b)$$

$$b = b - \alpha \frac{\partial}{\partial b} J(\vec{w}, b)$$

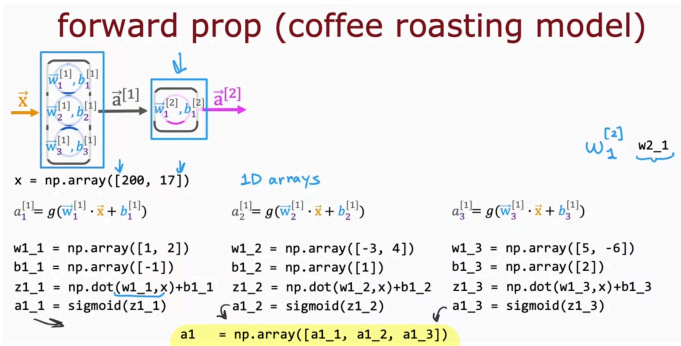
$$\rightarrow = \frac{1}{m} \sum_{i=1}^{m} \left[\left(f_{w,b}(x^{(i)}) - y^{(i)}\right) x_j^{(i)} \right] + \frac{\lambda}{m} w_j^{(i)}$$

$$= \frac{1}{m} \sum_{i=1}^{m} \left(f_{w,b}(x^{(i)}) - y^{(i)}\right)$$

C2



$$a_j^{[l]} = g(\vec{w}_j^{[l]} \cdot a^{[l-1]} + b_j^{[l]})$$



Numerical Roundoff Errors

Ex: Logistic Regression

$$a = g(z) = \frac{1}{1+e^{-z}}$$

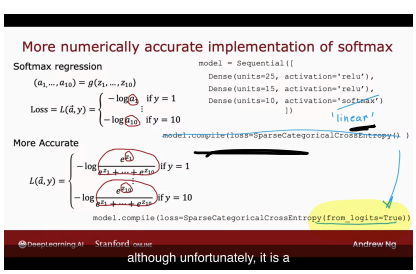
Logistic loss

$$loss = -y\log(a) - (1-y)\log(1-a)$$

More accurately:

$$loss = -y\log\left(\frac{1}{1+e^{-z}}\right) - (1-y)\log\left(1-\frac{1}{1+e^{-z}}\right)$$

logits



Multi-labeled Classification

1. Treat as separate classification problems
2. A NN to do multiple classification $\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$ car / bus / pedestrian

---

⑨ Regularization and bias/variance

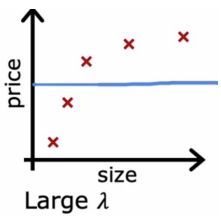
Model: $f_{\vec{w},b}(x) = w_1x + w_2x^2 + w_3x^8 + w_4x^{10} + b$

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

as λ bigger, model would try to keep w_j smaller.

but if $\lambda = 0$, there is no regularization → overfitting

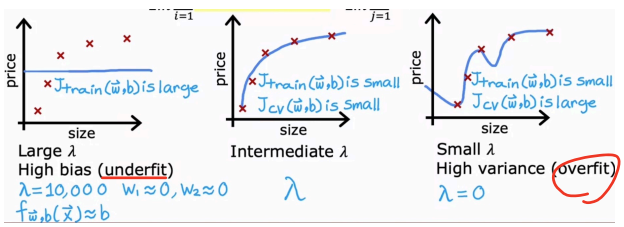
J_{train} is small, but J_{cv} is large



$\lambda = 10{,}000$, $w_1 \approx 0$, $w_2 \approx 0$

$f_{\vec{w},b}(x) \approx b$ ← high bias

J_{train} is large



Overfitting: J_{train} is small, J_{cv} is large

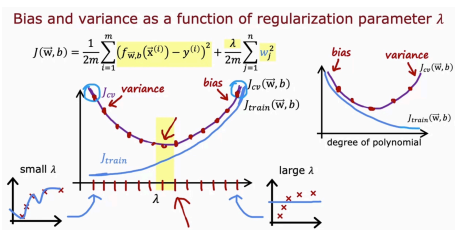
Underfitting: J_{train} is large, J_{cv} is more larger

choose the λ value by examining the J_{cv}

& if we have λ with multiple values → we train model with multiple w,b pairs then we choose the pair with lowest J_{cv}. apply the pair w,b to test error

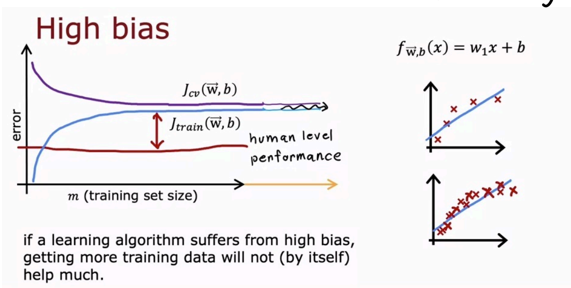
$\lambda \uparrow$, $w \downarrow$ → $J_{w,b} \approx b$ (high bias)

$\lambda \downarrow$, $w \uparrow$ → overfitted



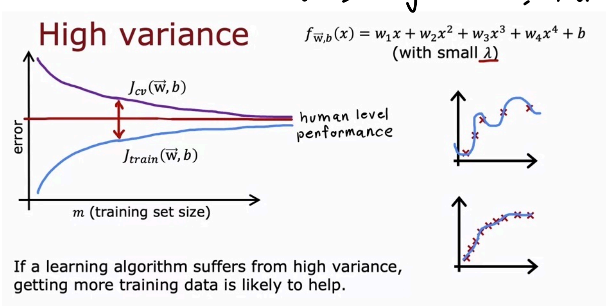
※ high bias / high variance

1. high bias → underfitting (是 algo 需要 improved, 增加 data 量不会改善 bias)



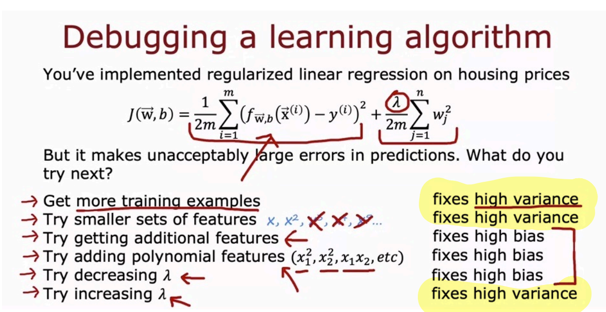
2. high variance → overfitting

lower λ → higher W, far away from high bias situation



※ Debugging the learning algorithm

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



没有减少 training examples 的 option

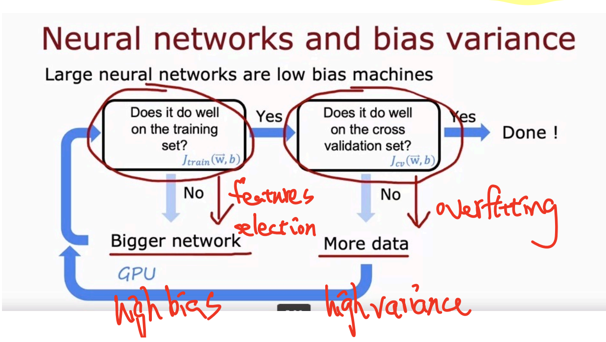
high bias → 主要是 model 如何学得更好的問題

① 希望 low bias → d↑, (λ↓)

② 多些 feature 让 model 能 fitting

③ polynomial 的 features 加大

→ 使 fitting 更 non-linear



※ NN and bias/variance

1. 通常大NN都能在小dataset中有小bias，不然就是NN要慢慢加大(e.g. 多一些layers)

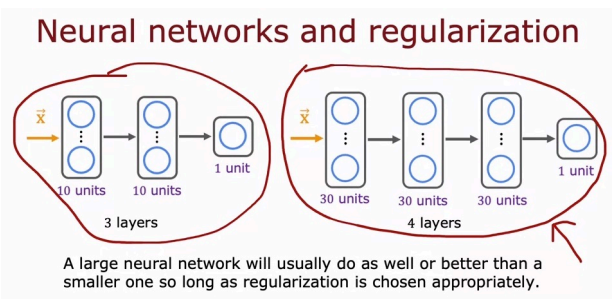
所以通常要solve的是high variance problem，不是大bias problem

2. performance：①先以human performance視為baseline，再看J_{train} & J_{CV}

3. 大NN不一定會有小bias，因為達成小bias是要fit好一些
   ① 多一些feature使model學得更好
   ② N個，刁N
   ③ 複雜一些的feature (non-linear)

4. 加大NN不一定好，在feature不夠精準時，計算太expensive

5. 怕使大NN來降低bias時引起high variance (overfitting) → 對大NN用Regularization



6. check underfitting (high bias)：看J_{train}與baseline差距
   check overfitting (high variance)：看J_{train}與J_{CV}差距

※ ML process

1. Error Analysis：

   ① Group the mis-classified examples
   將分錯的做grouping並找出共通點(eg. mis-spelling)

   ② 分錯的"少數幾個group"，可collect more data讓model去學 (examples太少)

   ③ spam email classification也可透過suspicious URLs之間的links做為feature來train URLs-based的spam emails classifier

⊛ the core idea from Bias/Variance

1. collect more data or not?
2. extract / minimize features?
3. do more/less on L2 Regularization λ?
4. Think more on model / data

⑤ Error Analysis only works on what we human are able to "observe"

- ※ Adding data on improving overfitting (high variance)

1. 針對 Error Analysis，針對分錯的"次族"collect more data 以 improve 錯誤率

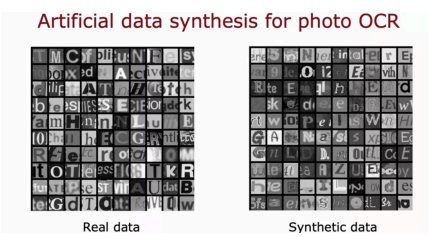
2. Data Augumentation: Modify existing examples to better train model (e.g. flip/distoration/rotate... on image)

⊙ The "distortion" should also be in the testing set!

⟶ help us know whether the distortion for training process is useful

3. Data synthesis: creating brand new examples

1. Generate the realistic data similar to real data

2. 

Focus more on "data" could be more efficient way to collaborate with existed advanced algorithms.

※. Transfer learning

1. Use the pre-trained model, but only train the final output layer params

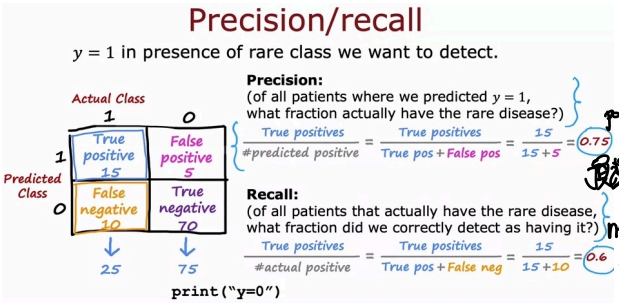
2. The dimension of our data and the pre-trained data should be the same.

3. Use the pretrained model, then train it entirely on our dataset

※ Skewed datasets

1. Problem: if train a model, it would be good at classifying the type of majority in the pre-trained data

2. Error metric: Precision / Recall



	實際的 1	實際的 0
預測的 pos	真的 pos (TP)	假的 (FP)
預測的 neg	假的 neg (FN)	真的 neg (TN)

衡量 algo 在 skewed dataset，可以從 precision, recall

$$\text{precision} = \frac{TP}{TP+FP} \qquad \text{recall} = \frac{TP}{TP+FN}$$

recall 的出現是為了防止 TP+FP=0 的 scenario

→ precision 和 recall 的 trade-off 很重要

High precision: classifier 真正識別出有病者中，真正病的機率高

High recall = $\frac{TP}{TP+FN}$ recall↑, 表示 FN↓, → classifier 將有病者誤判的機率變低

實際上有病但認為沒病

※ Decision Trees

1. How to choose features at each node?

→ Maximum purity

2. To estimate purity: "Entropy"

3. When to stop splitting? → avoid overfitting

keep the tree smaller

pre-defined

① 100% pure on classes

② When exceeding the depth of tree

③ When the improvements in purity score are below a threshold

④ When the number of example in one node is below a threshold

→ threshold ① max depth
② improvements of purity
③ #examples in nodes

4. Choose what features to split on ?

→ main goal: maximum purity !

Key: When to stop splitting !

Define a decision node!

5. Entropy as a measure of impurity

$$H(p_1) = -p_1 \log(p_1) - p_2 \log(p_2)$$

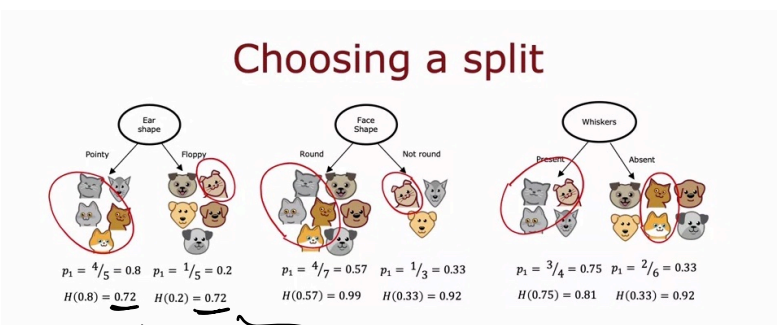
$$= -p_1 \log(p_1) - (1-p_1) \log((1-p_1))$$

6. Decision Nodes: Information Gain

1. The chose of decision nodes hope to reduce the entropy

2. Information Gain = Reduction of entropy

3.



$H(0.8) = -0.8\log(0.8) - 0.2\log(0.2)$

$= H(0.2) = 0.72$

左右两边 — weighted sum. weights 以该树的 # samples 决定

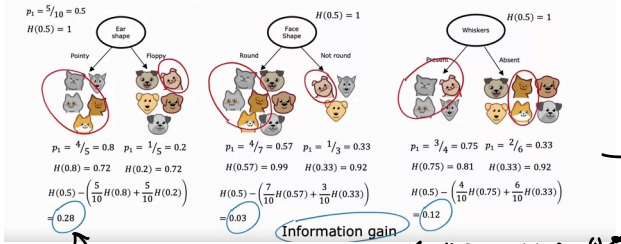
$\frac{5}{10}0.72 + \frac{5}{10}0.72 = 0.72$

原始的：cat, dog 各半：$H(0.5) = -0.5\log(0.5) - 0.5(\log 0.5)$

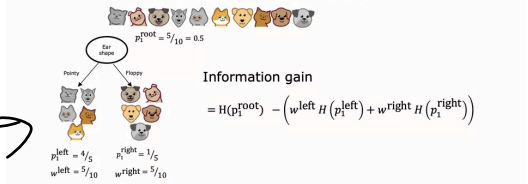
→ Information Gain = Reduction of entropy

$= H(0.5) - \left[\frac{5}{10}H(0.8) + \frac{5}{10}H(0.2)\right]$

Choosing a split



Information Gain



最大的 Information Gain：用於從 Root Node 分裂後最開始的 Decision Node

Decision Tree Learning

- Start with all examples at the root node : We get the original distribution
- Calculate information gain for all possible features, and pick the one with the highest information gain → 算可設的 entropy：用 weighted sum
- Split dataset according to selected feature, and create left and right branches of the tree
- 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 splits is less than threshold
  - When number of examples in a node is below a threshold

1. 100% 分類
2. max depth
3. class下的件 samples < threshold
4. Information Gain < threshold

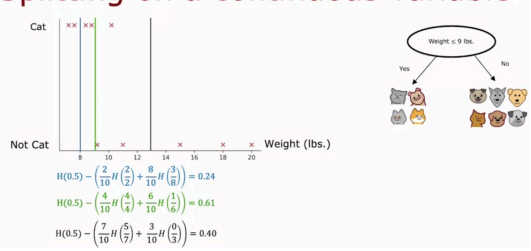
Categorical Feature Encoding

1. one-hot encoding: can be fed into NN, decision tree, logistic regression...
2. label encoding — let input takes on more discrete value
3. ordinal encoding
4. interval encoding

Encoding for continuous valued features

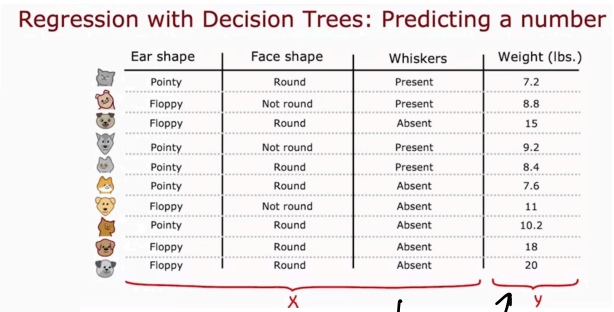
1. Like we said, one-hot encoding takes care of discrete-number features
2. Now, we need other methods to encode continuous-valued features
3. Try different threshold, and calculate different information gain → then get the threshold

Splitting on a continuous variable



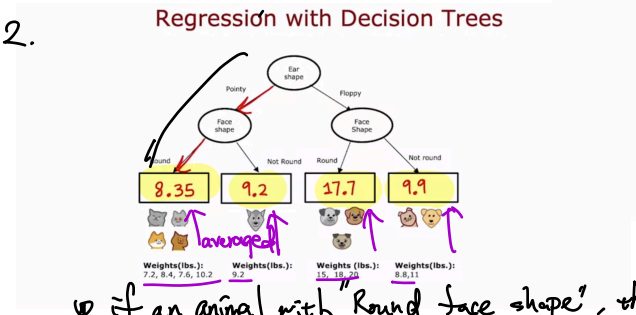
9 Use decision tree to predict a number rather than a "class"

1. Called Regression Tree



predict the weights of a given animal

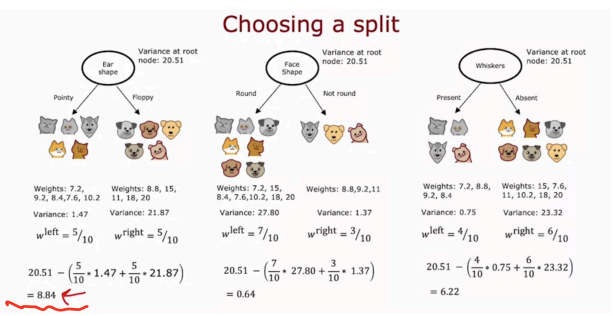
2.



If an animal with "Round face shape", the 8.35 is the averaged weights of training examples' weights

3. 若在決定 decision node 時，分下來因為沒有對象的 continuous valued features 導致無法計算 information gain 時 → 看 continuous values 的 variance

用 values 的 variance 來代 information gain，再依分裂數量做 weighted sum.



we get the "reduction in variance", then choose the subtree returns the "most reduction in variance"

跟 information gain 一樣，要選每個原始 entropy / variance 相減後 "最大" 者做為當下的 decision node. 因為 改善越多 (reduction in entropy/variance) 才是收斂的目標

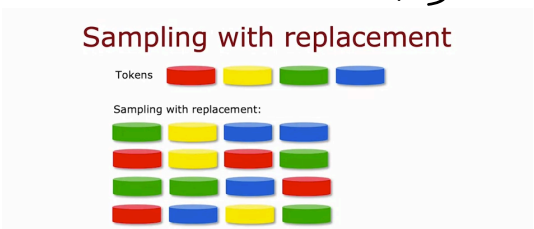
9 Using multiple decision trees: Random Forest

1. Why? Using single decision tree is sensitive to small changes in data

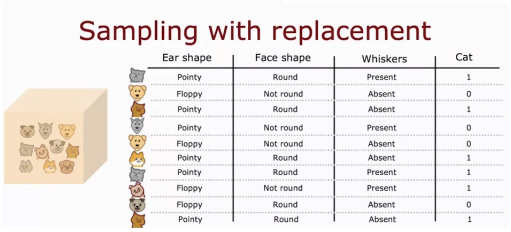
→ to be more robust, we need more decision tree to do ensemble

2. Build lots of decision tree to do tree ensemble

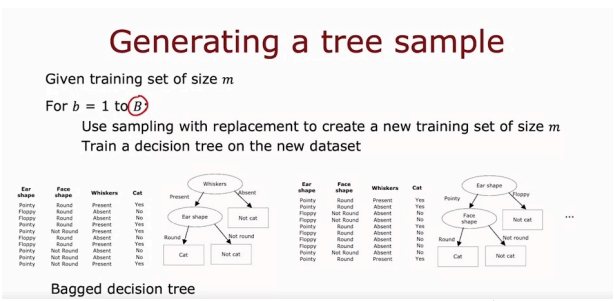
3. Tree Ensemble: Sampling with replacement : not get the same four tokes everytime



And, some samples may repeated.
We use this technique to create new training for ensemble voting



4.



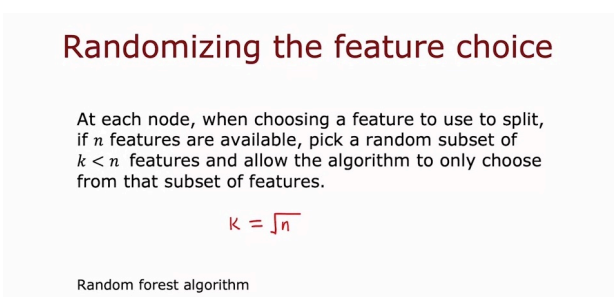
做B次 Sampling with Replacement，每次再train個 decision tree

B的大小? 64/228... B愈大可能到某個 threshold 就不會 better了，

↑ 根據 data 決定，沒有 default，但不會超過 1000. 計算量太大且不會比較好.

通常可能很多次 root node 或 root node 附近的 node 在多次的 Sampling with Replacement 都很像，除非更多次 data 抽完差異性很大

∴ 需要 randomize! 將原本可用的 features 只取某 subset 來做為 decision node 的條件



5. XGBoost: 讓model focus 在分得不好的 samples 上繼續 train，跟人一樣

建多個decision tree 做 ensemble

Boosted trees intuition

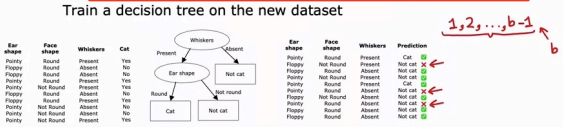
Given training set of size m

For b = 1 to B:

Use sampling with replacement to create a new training set of size m

But instead of picking from all examples with equal (1/m) probability, make it more likely to pick misclassified examples from previously trained trees

Train a decision tree on the new dataset



分得不好的 samples 会在下轮中被高机率被抽到

1, 2, ..., b-1

b

XGBoost (eXtreme Gradient Boosting)

- Open source implementation of boosted trees
- Fast efficient implementation
- Good choice of default splitting criteria and criteria for when to stop splitting
- Built in regularization to prevent overfitting
- Highly competitive algorithm for machine learning competitions (eg: Kaggle competitions)

不需自行做很多 sampling with replacement 的 dataset 来 train，XGBoost 会自行对 samples 动手脚

◎ 前述提到的 1. Sampling with Replacement 2. Randomizing the feature choice

都是 XGBoost 会用到的方法

◎ Decision Trees v.s. NN

1. DT 通常用在 structured data (like Tabular data, spreadsheet) with
   ① Categorical or Continuous-valued features
   ② Predict for classification or Regression
2. Interpretable：较小的 DT 比较 interpretable
3. Ensemble 而非一 DT 差在计算量较大
4. NN：可在 DT 的 structured data (spreadsheet-liked) worked well.
   和 un-structured data (all types of data)

Decision Trees vs Neural Networks

Decision Trees and Tree ensembles

- Works well on tabular (structured) data
- Not recommended for unstructured data (images, audio, text)
- Fast
- Small decision trees may be human interpretable

Neural Networks

- Works well on all types of data, including tabular (structured) and unstructured data
- May be slower than a decision tree
- Works with transfer learning
- When building a system of multiple models working together, it might be easier to string together multiple neural networks

◎ K-means Optimization objective

1. $c^{(i)}$: index of cluster (1, 2, 3, ..., k) to which example $x^{(i)}$ is currently assigned

   μ_k: cluster centroid k

   $\mu_{c^{(i)}}$: cluster centroid of cluster to which example $x^{(i)}$ has been assigned

Cost function:

$$J(c^{(1)},\dots c^{(m)}, \mu_1,\dots \mu_K) = \frac{1}{m}\sum_{i=1}^{m}\|x^{(i)} - \mu_{c^{(i)}}\|^2$$ Distortion Function

Cost function for K-means

$$J(c^{(1)},\dots,c^{(m)},\mu_1,..,\mu_K) = \frac{1}{m}\sum_{i=1}^{m}\|x^{(i)} - \mu_{c^{(i)}}\|^2$$

Repeat {

 # Assign points to cluster centroids

for i = 1 to m

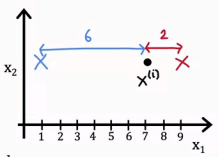
$c^{(i)}$:= index of cluster centroid closest to $x^{(i)}$

 # Move cluster centroids

for k = 1 to K

μ_k := average of points in cluster k

}



Initializing the cluster centroids

1. Random Initialization

Random initialization

For $i = 1$ to 100 { (50-1000)

Randomly initialize K-means. (k random examples)

Run K-means. Get $c^{(1)},\dots,c^{(m)},\mu_1,\ \mu_1,\dots,\ \mu_k$

Computer cost function (distortion)

$J(c^{(1)},\dots,c^{(m)},\mu_1,\ \mu_1,\dots,\ \mu_k)$

}

Pick set of clusters that gave lowest cost J

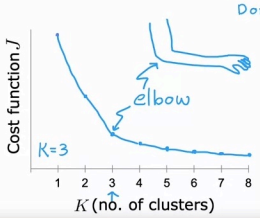
2. Choose the number of cluster centroids

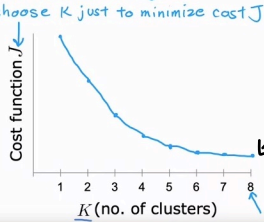
Choosing the value of K

Elbow method

the right "K" is often ambiguous

Don't choose K just to minimize cost J

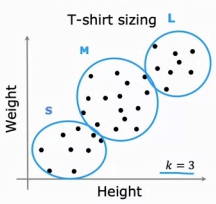


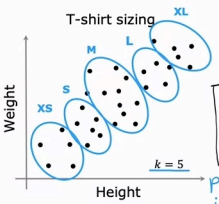


不能选 last one. 因为会越多 cluster, 通则基本 loss 会小

Choosing the value of K

Often, you want to get clusters for some later (downstream) purpose.
Evaluate K-means based on how well it performs on that later purpose.





practice lab: image compression!

Which of these statements are true about K-means? Check all that apply.

- [x] If you are running K-means with $K = 3$ clusters, then each $c^{(i)}$ should be 1, 2, or 3.

  Correct
  $c^{(i)}$ describes which centroid example (i) is assigned to. If $K = 3$, then $c^{(i)}$ would be one of 1,2 or 3 assuming counting starts at 1.

- [x] If each example x is a vector of 5 numbers, then each cluster centroid μ_k is also going to be a vector of 5 numbers.

  Correct
  The dimension of μ_k matches the dimension of the examples.

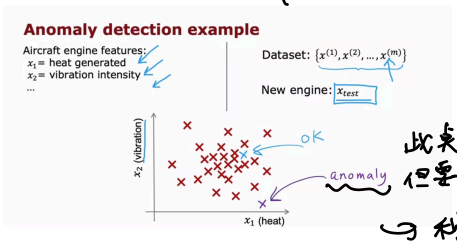
- [x] The number of cluster assignment variables $c^{(i)}$ is equal to the number of training examples.

  Correct
  $c^{(i)}$ describes which centroid example (i) is assigned to.

- [ ] The number of cluster centroids μ_k is equal to the number of examples.

① Anomaly Detection

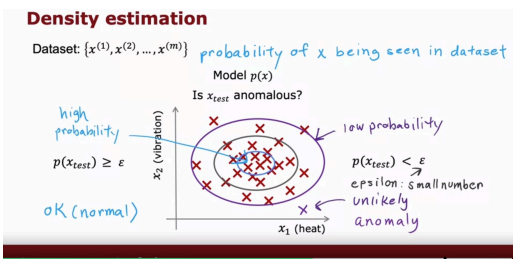
1. 異常檢測：從選定的 features 做 embedding (or vectors)，判定新的 sample 是不是異常 data



此案明顯不與大家在一起，但要如何決定這個 sample 是 anomaly

↳ 利用 Density Estimation

2. Density Estimation

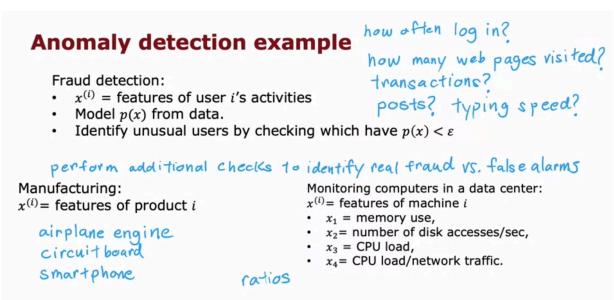


① 先以 training set 中找出常見的特徵做為 features，建立 probability model，(eg. x_1, x_2 做為 features；$x^{(1)} \sim x^{(m)}$ 都是 train set)

② 當新的 data 進來時，以建好的 probability model 計算 $p(x_{test})$，同時也有自訂一個 threshold ε

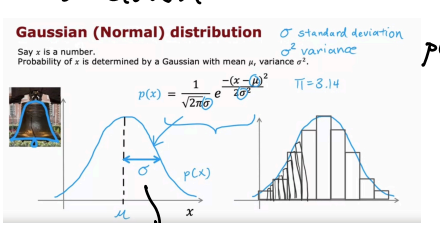
③ 若 $P(x_{test}) > \varepsilon$：Normal，else anomaly

Applications:



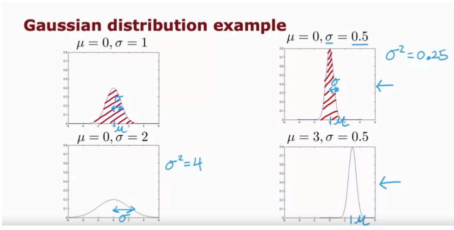
3. Methods

① Gaussian Distribution



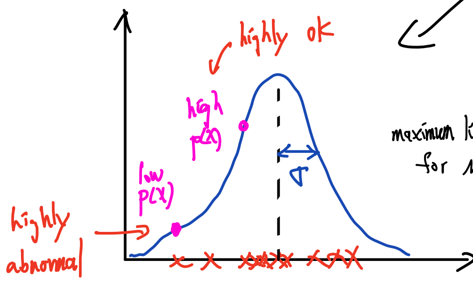
$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

σ↓，高凸，樣本愈集中



用在 Anomaly Detection:

以單一 feature 為例

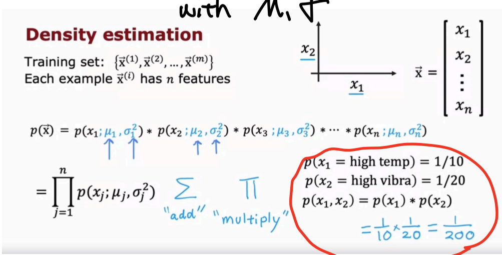


maximum likelihood, for μ, σ

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - \mu)^2$$

2. Density Estimation with μ, σ



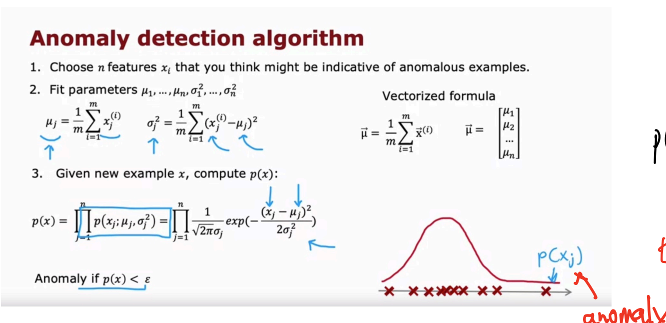
顯示若要 temp、vibra 同時發生的 prob 很低

當有很多 feature 時 → 針對每個 possible feature 都算 p(x)

$$p(\vec{x}) = p(x_1;\mu_1,\sigma_1^2)\,p(x_2;\mu_2,\sigma_2^2)\cdots p(x_n;\mu_n,\sigma_n^2)$$

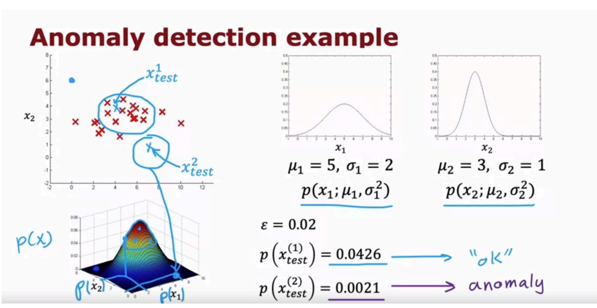
↑ 將每個特徵視為條件獨立

$$= \prod_{j=1}^{n} p(x_j;\mu_j,\sigma_j^2)$$

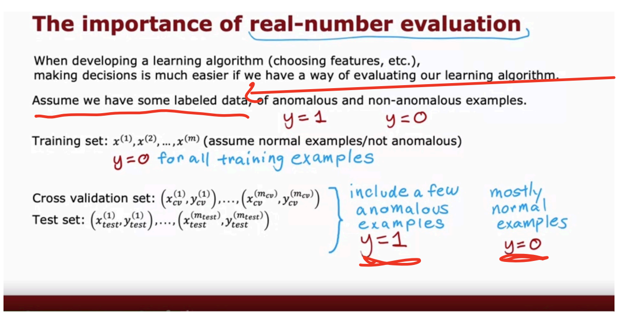


$$p(x) = \prod_{j=1}^{n} p(x_j;\mu_j,\sigma_j^2) = \prod_{j=1}^{n}\frac{1}{\sqrt{2\pi}\sigma_j} e^{\frac{-(x_j-\mu_j)^2}{2\sigma_j^2}}$$

當其中一個 p(xj) 很小時 → 代表為離群值，所有 p(xj) 相乘的結果也會超小 → p(x) < ε → anomaly



此時剩下的問題是 ε 應為多少較合適



根據收到的 prediction 做 labeling，經過一段時間可以做自己擁有一些 label 的 dataset

3. Evaluate the algorithm

Aircraft engines monitoring example

10000 good (normal) engines — 2 to 50
20 flawed engines (anomalous) — y=1
2 — y=0

Training set: 6000 good engines — train algorithm on training set

CV: 2000 good engines ($y = 0$) — 10 anomalous ($y = 1$)
use cross validation set — tune ε — tune x_j

Test: 2000 good engines ($y = 0$), — 10 anomalous ($y = 1$)

Alternative: No test set — Use if very few labeled anomalous examples
Training set: 6000 good engines 2 — higher risk of overfitting
CV: 4000 good engines ($y = 0$), 20 anomalous ($y = 1$)
tune ε tune x_j

1. 以之前20個 anomaly sample 為例，分10個到 cross-validation，可以在 train 完後看 predict CV 時，ε 應設多小能盡可能辨識 anomaly 找出來。
2. ε 測完後，再到 test set 跑
3. 如果只有 2 個 anomaly，那就不要 test 組了，只建 train 和 CV，且把 anomaly 都放在 CV 讓我們去 tune ε，這是唯一的方法且很可能會造成 overfitting.

Algorithm evaluation

course 2 week 3 skewed datasets

Fit model $p(x)$ on training set $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$
On a cross validation/test example x, predict

$$y = \begin{cases} 1 & \text{if } p(x) < \varepsilon \text{ (anomaly)} \\ 0 & \text{if } p(x) \geq \varepsilon \text{ (normal)} \end{cases}$$

10
2000

Possible evaluation metrics:
- True positive, false positive, false negative, true negative
- Precision/Recall
- F_1-score

Use cross validation set to choose parameter ε

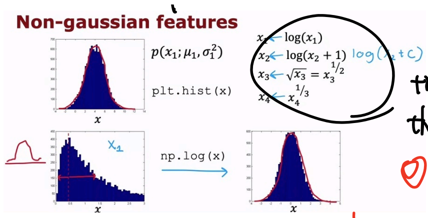
※ Anomaly Detection v.s. Supervised Learning

Anomaly detection	vs.	Supervised learning
Very small number of positive examples ($y = 1$). (0-20 is common). Large number of negative ($y = 0$) examples. $p(x)$ $y=1$		Large number of positive and negative examples. 20 positive examples
Many different "types" of anomalies. Hard for any algorithm to learn from positive examples what the anomalies look like; future anomalies may look nothing like any of the anomalous examples we've seen so far.		Enough positive examples for algorithm to get a sense of what positive examples are like, future positive examples likely to be similar to ones in training set.
Fraud 用於一直有不同變形或 features 的分類 ② 變動性/更新"高"		Spam 用於已經改變 features 的 classification task

Anomaly detection	vs.	Supervised learning
→ Fraud detection		→ Email spam classification
→ Manufacturing - Finding new previously unseen defects in manufacturing.(e.g. aircraft engines)		→ Manufacturing - Finding known previously seen defects $y=1$ scratches
→ Monitoring machines in a data center		→ Weather prediction (sunny/rainy/etc.)
⋮		→ Diseases classification
		⋮
features "Updating"		features "known"

※ Anomaly Detection: Choose the features

1. Supervised Learning: 一些"不重要或缺少一些不是很重要的 feature 並不會對 model 的 performance 帶來很大影響，因為 supervised 是有"label"可做 optimization
2. 但在無 label 的 condition 下，要有"對的且重要的 features"才能讓 algo 分得好或找到較好的 rule
3. So, carefully choose features & find important features is important in Anomaly Detection
4. We hope the distribution of data becomes more "Gaussian"
   - Make the non-Gaussian features more Gaussian
     ↓



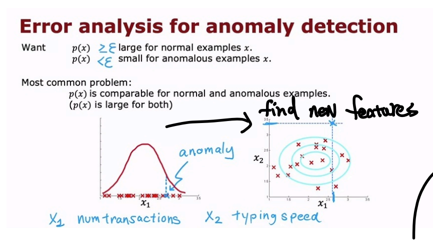
transform X to make the distribution more Gaussian

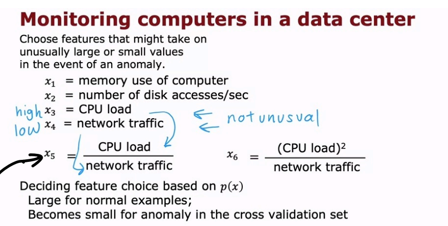
① If we make the Gaussian transformation method on train set, then we have to apply the same transformation on test and val set.

5. Error Analysis for Anomaly Detection

▷ If the trained model doesn't work well on val set, we have to apply error analysis.

②

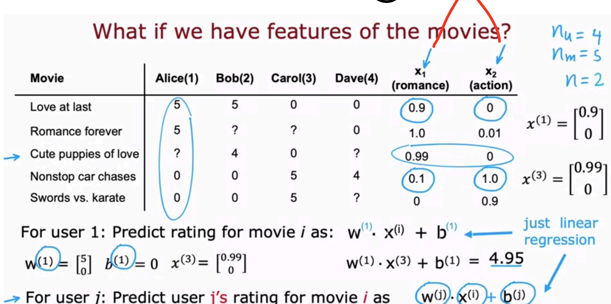




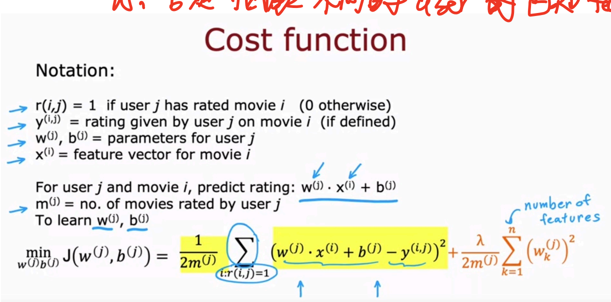
Create new feature

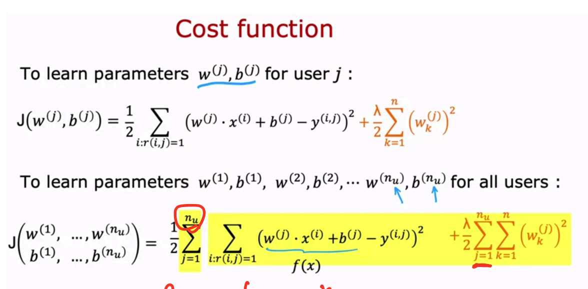
※ Collaborative Filtering

额外的 movie features



W, b是根据不同的 user 的已知 feature 下去 fit 出来的





for each user

↑ with per-item feature

what if we don't have advanced features?

③ Collaborative filtering algorithm

Problem motivation

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	x_1 (romance)	x_2 (action)
Love at last	5	5	0	0	?	?
Romance forever	5	?	?	0	?	?
Cute puppies of love	?	4	0	?	?	?
Nonstop car chases	0	0	5	4	?	?
Swords vs. karate	0	0	5	?	?	?

$w^{(1)}=\begin{bmatrix}5\\0\end{bmatrix}$, $w^{(2)}=\begin{bmatrix}5\\0\end{bmatrix}$, $w^{(3)}=\begin{bmatrix}0\\5\end{bmatrix}$, $w^{(4)}=\begin{bmatrix}0\\5\end{bmatrix}$

$b^{(1)}=0$, $b^{(2)}=0$, $b^{(3)}=0$, $b^{(4)}=0$

using $w^{(j)}\cdot x^{(i)}+b^{(j)}$

$w^{(1)}\cdot x^{(1)}\approx 5$

$w^{(2)}\cdot x^{(1)}\approx 5$

$w^{(3)}\cdot x^{(1)}\approx 0$

$w^{(4)}\cdot x^{(1)}\approx 0$

$\rightarrow \quad x^{(1)}=\begin{bmatrix}1\\0\end{bmatrix}$

超级 $w^{(2)}w^{(3)}$ 那次对 情况也都会给 $x^{(1)}$ 相似

为啥 linear regression 的 fitting

Cost function

Given $w^{(1)}, b^{(1)}, w^{(2)}, b^{(2)}, \cdots, w^{(n_u)}, b^{(n_u)}$

to learn $x^{(i)}$:

$$J(x^{(i)})=\frac{1}{2}\sum_{j:r(i,j)=1}\left(w^{(j)}\cdot x^{(i)}+b^{(j)}-y^{(i,j)}\right)^2+\frac{\lambda}{2}\sum_{k=1}^{n}\left(x_k^{(i)}\right)^2$$

→ To learn $x^{(1)}, x^{(2)}, \cdots, x^{(n_m)}$:

$$J(x^{(1)}, x^{(2)}, \ldots, x^{(n_m)})=\frac{1}{2}\sum_{i=1}^{n_m}\sum_{j:r(i,j)=1}\left(w^{(j)}\cdot x^{(i)}+b^{(j)}-y^{(i,j)}\right)^2+\frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}\left(x_k^{(i)}\right)^2$$

Collaborative Filtering

Collaborative filtering

	$j=1$ Alice	$j=2$ Bob	$j=3$ Carol
$i=1$ Movie1	5	5	?
$i=2$ Movie2	?	2	3

Cost function to learn $w^{(1)}, b^{(1)}, \cdots w^{(n_u)}, b^{(n_u)}$:

$$\min_{w^{(1)},b^{(1)},\ldots,w^{(n_u)},b^{(n_u)}}\frac{1}{2}\sum_{j=1}^{n_u}\sum_{i:r(i,j)=1}\left(w^{(j)}\cdot x^{(i)}+b^{(j)}-y^{(i,j)}\right)^2+\frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}\left(w_k^{(j)}\right)^2$$

Cost function to learn $x^{(1)}, \cdots, x^{(n_m)}$:

$$\min_{x^{(1)},\ldots,x^{(n_m)}}\frac{1}{2}\sum_{i=1}^{n_m}\sum_{j:r(i,j)=1}\left(w^{(j)}\cdot x^{(i)}+b^{(j)}-y^{(i,j)}\right)^2+\frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}\left(x_k^{(i)}\right)^2$$

Put them together:

$$\min_{\substack{w^{(1)},\ldots,w^{(n_u)}\\ b^{(1)},\ldots,b^{(n_u)}\\ x^{(1)},\ldots,x^{(n_m)}}}J(w,b,x)=\frac{1}{2}\sum_{(i,j):r(i,j)=1}\left(w^{(j)}\cdot x^{(i)}+b^{(j)}-y^{(i,j)}\right)^2+\frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}\left(w_k^{(j)}\right)^2+\frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}\left(x_k^{(i)}\right)^2$$

代表部分

Gradient Descent

collaborative filtering

Linear regression (course 1)

repeat {

$w_i=w_i-\alpha\frac{\partial}{\partial w_i}J(w,b)$ $\quad w_i^{(j)}=w_i^{(j)}-\alpha\frac{\partial}{\partial w_i^{(j)}}J(w,b,x)$

$b=b-\alpha\frac{\partial}{\partial b}J(w,b)$ $\quad b^{(j)}=b^{(j)}-\alpha\frac{\partial}{\partial b^{(j)}}J(w,b,x)$

$x_k^{(i)}=x_k^{(i)}-\alpha\frac{\partial}{\partial x_k^{(i)}}J(w,b,x)$

}

parameters w, b, x $\quad$ x is also a parameter

Cost function for binary application

Previous cost function:

$$\frac{1}{2}\sum_{(i,j):r(i,j)=1}\left(w^{(j)}\cdot x^{(i)}+b^{(j)}-y^{(i,j)}\right)^2+\frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}\left(x_k^{(i)}\right)^2+\frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}\left(w_k^{(j)}\right)^2$$

f(x)

Loss for binary labels $y^{(i,j)}$: $f_{(w,b,x)}(x)=g(w^{(j)}\cdot x^{(i)}+b^{(j)})$

$$L\left(f_{(w,b,x)}(x), y^{(i,j)}\right)=-y^{(i,j)}\log\left(f_{(w,b,x)}(x)\right)-\left(1-y^{(i,j)}\right)\log\left(1-f_{(w,b,x)}(x)\right)$$ Loss for single example

$$J(w,b,x)=\sum_{(i,j):r(i,j)=1}L\left(f_{(w,b,x)}(x), y^{(i,j)}\right)$$ cost for all examples

$g(w^{(j)}\cdot x^{(i)}+b^{(j)})$

Mean Normalization

Users who have not rated any movies

Movie	Alice(1)	Bob (2)	Carol (3)	Dave (4)	Eve (5)	
Love at last	5	5	0	0	?	0
Romance forever	5	?	?	0	?	0
Cute puppies of love	?	4	0	?	?	0
Nonstop car chases	0	0	5	4	?	0
Swords vs. karate	0	0	5	?	?	0

变成 matrix

$$\begin{bmatrix}5&5&0&0&?\\5&?&?&0&?\\?&4&0&?&?\\0&0&5&4&?\\0&0&5&0&?\end{bmatrix}$$

mean $(5+5+0+0)/4=2.5$

$(5+0)/2=2.5$

$(4+0)/2=2$

$(0+0+5+4)/4=2.25$

$(0+0+5+0)/4=1.25$

$$\rightarrow\begin{bmatrix}2.5\\2.5\\2\\2.25\\1.25\end{bmatrix}=\mu$$

$$\min_{\substack{w^{(1)},\ldots,w^{(n_u)}\\ b^{(1)},\ldots,b^{(n_u)}\\ x^{(1)},\ldots,x^{(n_m)}}}\frac{1}{2}\sum_{(i,j):r(i,j)=1}\left(w^{(j)}\cdot x^{(i)}+b^{(j)}-y^{(i,j)}\right)^2+\frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}\left(w_k^{(j)}\right)^2+\frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}\left(x_k^{(i)}\right)^2$$

$w^{(5)}=\begin{bmatrix}0\\0\end{bmatrix}$ $\quad b^{(5)}=0$ $\quad w^{(5)}\cdot x^{(i)}+b^{(5)}$

$$X-\mu=\begin{bmatrix}2.5&2.5&-2.5&-2.5&?\\2.5&?&?&-2.5&?\\?&?&-2&?&?\\-2.25&-2.25&2.25&1.75&?\\-1.25&-1.25&3.75&-1.25&?\end{bmatrix}$$

For user j, on movie i predict:

$$w^{(j)}x^{(i)}+b^{(j)}+\mu$$

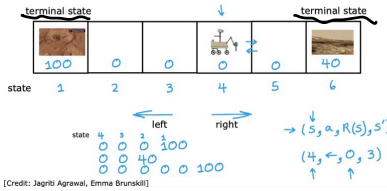
mean normalization: make algo run a little bit faster and gives reasonable predictions

Reinforcement Learning

1. "What to do", not "how to do it".
   ↑ algorithm has to learn itself

2.

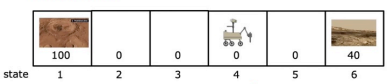
Mars Rover Example



← (current state, next action, reward of current state, next state)

3. Return:

Return



Return $= 0 + (0.9)0 + (0.9)^2 0 + (0.9)^3 100 = 0.729 \times 100 = 72.9$

Return $= R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots$ (until terminal state)

Discount Factor $\gamma = 0.9 \quad 0.99 \quad 0.999$ ← the discount factor should be close to 1

$\gamma = 0.5$

Return $= 0 + (0.5)0 + (0.5)^2 0 + (0.5)^3 100 = 12.5$

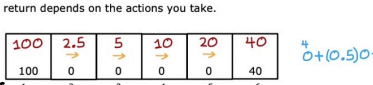
weighted by the discount factor

(在financial的scenario中)

Example of Return

$\gamma = 0.5$

The return depends on the actions you take.



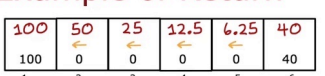
$0 + (0.5)0 + (0.5)^2 40 = 10$

$4: (0.5)^0 0 + (0.5)^1 0 + (0.5)^2 0 + (0.5)^3 100 = 12.5$

$3: (0.5)^0 0 + (0.5)^1 0 + (0.5)^2 0 + (0.5)^3 40 = 5$

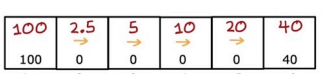
→ take the action based on what state we are at:

Example of Return

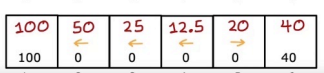


$\gamma = 0.5$

The return depends on the actions you take.



$0 + (0.5)0 + (0.5)^2 40 = 10$



$0 + (0.5)40 = 20$

discount factor

$\gamma = 0.5$

when gamma close to 1:

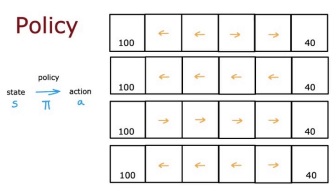
means the algorithm is "less impatient"

→ more patient

to get approximate to the real reward

Reward could be negative rewards!

Policy



$\pi(5) = a$
$\pi(2) = \leftarrow$
$\pi(3) = \leftarrow$
$\pi(4) = \leftarrow$
$\pi(5) = \rightarrow$

A policy is a function $\pi(s) = a$ mapping from states to actions, that tells you what action a to take in a given state s.

The goal of reinforcement learning

Find a policy π that tells you what action $(a = \pi(s))$ to take in every state (s) so as to maximize the return.

	Mars rover	Helicopter	Chess
states	6 states	position of helicopter	pieces on board
actions	← →	how to move control stick	possible move
rewards	100, 0, 40	+1, −1000	+1, 0, −1
discount factor γ	0.5	0.99	0.995
return	$R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots$	$R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots$	$R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots$
policy π		Find $\pi(s) = a$	Find $\pi(s) = a$

the "first reward" comes from "current state" without discount!

Markov Decision Process (MDP)

1. future only depends on current state, not have anything prior.

choose best "a" based on $\pi(s)$

State-action value function

State action value function (Q-function)

$Q(s,a)$ = Return if you
- start in state s.
- take action a (once).
- then behave optimally after that.

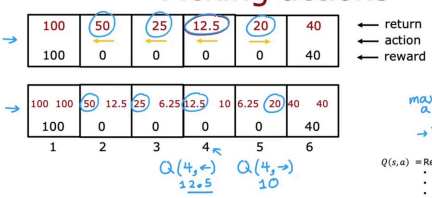
$Q(s,a)$ returns the value by taking the action "a" with state "s"

$Q(2,\rightarrow) = 12.5$
$0 + (0.5)0 + (0.5)^2 0 + (0.5)^3 100$
$Q(2,\leftarrow) = 50$
$0 + (0.5)100$
$Q(4,\leftarrow) = 12.5$
$0 + (0.5)0 + (0.5)^2 0 + (0.5)^3 100$

terminal state with reward

goal: get the maximum of Q, so we have to pick the actions.

Picking actions



$Q(4,\leftarrow)$ 12.5 $Q(4,\rightarrow)$ 10

$\max_a Q(s,a) \rightarrow \pi(s) = a$

$Q(s,a)$ = Return if you
- start in state s.
- take action a (once).
- then behave optimally after that.

The best possible return from state s is $\max_a Q(s,a)$.
The best possible action in state s is the action a that gives $\max_a Q(s,a)$.

Q^* Optimal Q function

Bellman Equation: Help us to compute the state action value function

Bellman Equation

$Q(s,a)$ = Return if you
- start in state s.
- take action a (once).
- then behave optimally after that.

R(1)=100 R(2)=0 ... R(6)=40

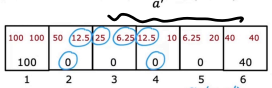
s : current state
a : current action
s' : state you get to after taking action a
a' : action that you take in state s'

$R(s)$ = reward of current state

Bellman Equation

$$Q(s,a) = R(s) + \gamma \max_{a'} Q(s',a')$$

$Q(s,a) = R(s)$



$s=2$ $a=\rightarrow$ $s'=3$

$Q(2,\rightarrow) = R(2) + 0.5 \max_{a'} Q(3,a')$
$= 0 + (0.5)25 = 12.5$

$s=4$ $a=\leftarrow$ $s'=3$

$Q(4,\leftarrow) = R(4) + 0.5 \max_{a'} Q(3,a')$
$= 0 + (0.5)25 = 12.5$

Explanation of Bellman Equation

$Q(s,a)$ = Return if you
- start in state s.
- take action a (once).
- then behave optimally after that.

$s \rightarrow s'$

The best possible return from state s' is $\max_{a'} Q(s',a')$

Recursion

$$Q(s,a) = R(s) + \gamma \max_{a'} Q(s',a')$$

Reward you get right away

Return from behaving optimally starting from state s'.

$Q(s,a) = R_1 + \gamma[R_2 + \gamma R_3 + \gamma^2 R_4 + \cdots]$

$R_1 + \gamma R_2 + \gamma^3 R_3 + \gamma^3 R_4 + \cdots$

$Q(2,\rightarrow) = R(2) + \max_{a'} Q(3,a')$

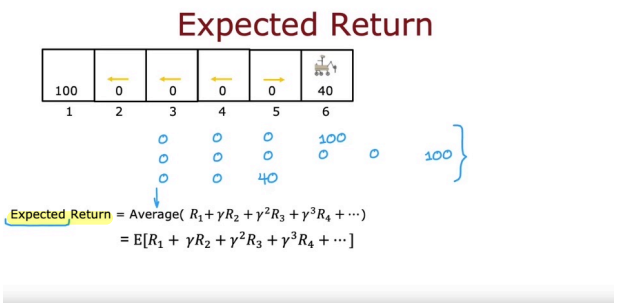
$Q(s,a) = R(s) + \gamma \max_{a'} Q(s',a')$

a' we have to calculate the maximum with state s' and its

※ Random/Stochastic Environment

possible actions a'

1. In the natural/random environment, the real move of target is not 100% guaranteed, for example: The wheel of robot may slip to the opposite direction.
2. The goal: Choose a policy "π" to maximum the average/expected discounted rewards.



3. The modified Bellman Equation

$$Q(s,a) = R(s) + \gamma E(\max_{a'} Q(s',a'))$$

2, ←

1 or 3

4. For example:

For this problem, $\gamma = 0.25$. The diagram below shows the return and the optimal action from each state. Please compute Q(5, ←).

100	25	6.25	2.5	10	40	← return
100	0	0	0	0	40	← action
1	2	3	4	5	6	← reward

$Q(5,\leftarrow)=?$

$Q(5,\leftarrow) = R(5) + \gamma R(4,\rightarrow)$

$= R(5) + \gamma(R(4) + \gamma R(5,\rightarrow))$

$= R(5) + \gamma R(4) + \gamma^2 R(5) + \gamma^3 R(6)$

$= 0 + 0 + 0 + (0.25)^3 * 40 = 0.0625 * \frac{1}{4} * 40 = 0.625$ ※

-※ Continuous state spaces

Discrete vs Continuous State

Discrete State:

Continuous State:

$s = \begin{bmatrix} x \\ y \\ \theta \\ \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}$ ← position at (x, y)
← orientation ∈ (0, 360)
← velocity in x, y direction
and in orientation

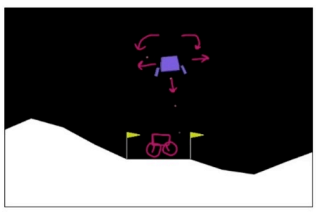
In Helicopter (3D 空拍机模像)

→ position: x, y, z

velocity: 左右旋转 (roll)
前.後傾斜 (pitch)
左.右偏斜 (yaw)

→ $\begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \omega \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\omega} \end{bmatrix}$

Lunar Lander



actions:
- do nothing
- left thruster
- main thruster
- right thruster

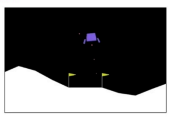
$s = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \\ \theta \\ \dot{\theta} \\ l \\ r \end{bmatrix}$ 0 or 1 ← ←

whether "left" or "right" touch the ground

Lunar Lander Problem

Learn a policy π that, given

$s = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \\ \theta \\ \dot{\theta} \\ l \\ r \end{bmatrix}$



picks action $a = \pi(s)$ so as to maximize the return.

$\gamma = 0.985$

$\begin{bmatrix} \dot{\theta} \\ \dot{w} \end{bmatrix}$

Given "s", use "π" to pick optimal "a" to maximize the return

Learn the state-value function for continuous states space

Deep Reinforcement Learning

$\vec{x} = \begin{bmatrix} s \\ a \end{bmatrix} \begin{bmatrix} x \\ y \\ \theta \\ \vdots \\ 1 \\ 0 \\ 0 \end{bmatrix}$ → ⋮ (64 units) → ⋮ (64 units) → ○ (1 unit) → $Q(s,a)$

$x \to y$

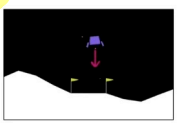
y

12 inputs

In a state s, use neural network to compute

$Q(s, \text{nothing})$, $Q(s, \text{left})$, $Q(s, \text{main})$ $Q(s, \text{right})$

Pick the action a that maximizes $Q(s,a)$



In the training data:

$Q(s,a) = R(s) + \gamma \max_{a'} Q(s', a')$

at first, the params of Q is random guess

x y

$f(x) \approx y$

to construct our training data:

if we don't have the π (good policy) yet,

take actions randomly!

→ then apply Bellman equation to improve the estimation of Q function

Bellman Equation

$Q(s,a) = R(s) + \gamma \max_{a'} Q(s', a')$

x y

random guess

$f_{w,B}(x) \approx y$

$(s, a, R(s), s')$

$y^{(1)} = R(s^{(1)}) + \gamma \max_{a'} Q(s'^{(1)}, a')$

$y^{(2)} = R(s^{(2)}) + \gamma \max_{a'} Q(s'^{(2)}, a')$

$(s^{(1)}, a^{(1)}, R(s^{(1)}), s'^{(1)})$ ←

$(s^{(2)}, a^{(2)}, R(s^{(2)}), s'^{(2)})$ ←

$(s^{(3)}, a^{(3)}, R(s^{(3)}), s'^{(3)})$ ←

$x^{(1)} = (s^{(1)}, a^{(1)})$ $y^{(1)}$

$x^{(2)} = (s^{(2)}, a^{(2)})$ $y^{(2)}$

$x^{(10,000)}$ $y^{(10,000)}$

Learning Algorithm

Initialize neural network randomly as guess of $Q(s,a)$.

Repeat {

Take actions in the lunar lander. Get $(s, a, R(s), s')$.

Store 10,000 most recent $(s, a, R(s), s')$ tuples.

Replay Buffer

Train neural network:

Create training set of 10,000 examples using

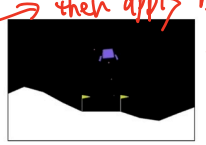
$x = (s, a)$ and $y = R(s) + \gamma \max_{a'} Q(s', a')$

Train Q_{new} such that $Q_{new}(s,a) \approx y$.

Set $Q = Q_{new}$.

$f_{w,B}(x) \approx y$

x, y $x^{(1)}, y^{(1)}$ ⋮ $x^{(10000)}, y^{(10000)}$



then apply Bellman equation to optimize the params in order to construct more precise Q

Algorithm refinement: ε-greedy policy

↖ Which action should we choose?

2. 在 random initialize 的 circumstance 下，可能有些 action 的 Q 永遠都很小，但這種 action 實際上需要加進來 train

→ try out something that might not be the best idea

How to choose actions while still learning?

In some state s

Option 1:

Pick the action a that maximizes $Q(s,a)$.

Option 2:

With probability 0.95, pick the action a that maximizes $Q(s,a)$. Greedy, "Exploitation"

With probability 0.05, pick an action a randomly. "Exploration"

ε–greedy policy ($\varepsilon = 0.05$)

0.95

0.95 greedy

Q(s,main) is low

↑ a

Start ε high → decrease the ε

1.0 → 0.01

Gradually decrease → less greedy → greedy

→ ε get lower = get greedy

① Mini-batch & Soft Update ① gradual updates ② converge better

① 加速

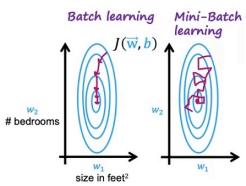
② 避免 local optima

< Mini-batch : ① faster when GD ② only for large dataset because of some existing outliers/noises

③ 大的 dataset 才能在 noise 造成的 loss 變大時，後續校正回來

Mini-batch

x	y
2104	400
1416	232
1534	315
852	178
...	...
3210	870



Learning Algorithm

Initialize neural network randomly as guess of $Q(s,a)$

Repeat {

Take actions in the lunar lander. Get $(s, a, R(s), s')$.

Store 10,000 most recent $(s, a, R(s), s')$ tuples.

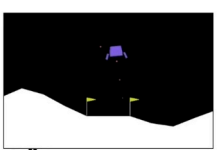
Train model: 1,000

Create training set of 10,000 examples using

$x = (s,a)$ and $y = R(s) + \gamma \max_{a'} Q(s', a')$

Train Q_{new} such that $Q_{new}(s,a) \approx y$.

Set $Q = Q_{new}$.



Replay Buffer → to Replay Buffer into mini-batches

$x^{(1)}, y^{(1)}$

$x^{(1000)}, y^{(1000)}$

↳ 因為 mini-batch 有時會因 samples 的不同，造成該次 loss 變大，此時步伐 $Q = Q_{new}$ 是不理想的，不能馬上 replace

∴ 需要 soft update → make gradual changes to Q

Soft Update

Set $Q = Q_{new}$.

W, B ← W_{new}, B_{new}

$W = 0.01 W_{new} + 0.99 W_{old}$

$B = 0.01 B_{new} + 0.99 B_{old}$

→ update gradually, converge better

Limitations of Reinforcement Learning

- Much easier to get to work in a simulation than a real robot!
- Far fewer applications than supervised and unsupervised learning.
- But … exciting research direction with potential for future applications.