Topic 4 - Model improvement

Overfitting and underfitting

- **Overfitting**: The line fits perfectly, but it's not quite right, it's got a high variance, it's overly-complex
- **Underfitting**: is an overly simple, and technical term for this is saying it has a high bias.

We're trying to make our **model generalizable** that fits the data very well, but it's **generalizable to new data**

Bias-variance curve

- If we want to **evaluate a machine learning solution to be generalizable to new data**, to be a solution that just fits, we could use the bias-variance curve
- Now, the bias-variance curve allows us to plot along the **x-axis** some **measure of model complexity**.The **y-axis measures the error**
- If we plot a graph **after testing,** The test results doesn't approach the predicted values from Training the error due to overfitting

Ensure model is generalisable

1. Reduce the number of features
   - Manually select which features
   - Use model selection algorithm (e.g. cross-validation)
2. Regularisation
   - Keep all features. but **reduce the values of parameter** θ_j
   - Works well when have a lot of features

Regularisation

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 \leftarrow \theta = [1.8,\ 4.2, -2,\ 0.5, -0, 1]^T \qquad \sum \theta^2 \simeq 25$

$\theta_0 + \theta_1 x + \theta_2 x^2 \qquad \leftarrow \theta = [-0.1,\ 1.1, -0.1]^T \qquad \sum \theta^2 \simeq 1.2$

- A method of penalizing complexity in our Machine Learning model
- Regularization: add a penalty to the loss function based on complexity
  - e.g. based on θ^2 (L2 regularization)
  - or $|\theta|$ (L1 regularization)

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

Regularization hyperparameter λ

If λ too big → algorithm underfits (θ very small)
If λ too small → algorithm overfits (θ can be very large)

Gradient Descent and Regularisation

Randomly initialize θ, then loop:

1. Calculate J(θ)
2. Update

   $\theta_0^{new} = \theta_0 - \frac{\alpha}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})$ Do not regularize θ_0

   $\theta_j^{new} = \theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} - \lambda\theta_j$ $for\ j = 1,\dots n$

3. End Loop when convergent, i.e. $J(\theta) \simeq J(\theta^{new})$

Cross-validation

Train-test separation

To avoid overfitting is important that the **data we use to train** an algorithm is **not the same** as the **data we use to test** it

N-fold cross-validation

Example:

1. Split our data up into n sections, in this case four. We take one of those, we put it aside, train on all the remaining ones and test on the one that is kept aside
2. train on the remaining stuff, and then test on the data that was taken out, we repeat that four times
3. Then we can take the error value that we get, or the accuracy and take the average over all of them to get our overall algorithm accuracy, or error on n-fold cross-validation.

- Works well for evaluating algorithms with **fixed parameter** e.g. k=1, distance=Eculidean
- How do we evaluate a set of parameters?

Bayesian classification

Probabilistic modeling is a way of doing machine learning that allows us to deal with **uncertainty**

```
P(good graduate job) = 0.5
Prior probability

P(good graduate job | attend networking events) = 0.8
Posterior probability
```

- We want to calculate the posterior probability of an event, given the observed evidence, and that can be difficult to do directly.
- There is a way to calculate this probability indirectly, using a **generative model** of the **likelihood** that a certain outcome will lead to a particular observation

Bayes' Theorem

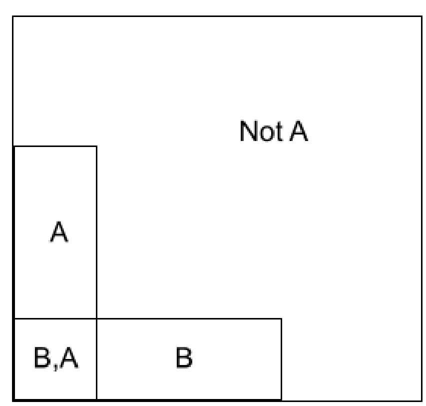
- Posterior: Probability of event given evidence
- Likelihood: Likelihood evidence generated by event
- Prior: What we believed about event beforehand
- Marginal Probability: Normalize constant, ensures posterior sums to 1 over all events

$$Posterior = \frac{Likelihood \cdot Prior}{Marginal\ Probability}$$

$$P(Y|X) = \frac{P(X|Y) \cdot P(Y)}{P(X)}$$

Probability Rules

$P(\overline{A}) = 1 - P(A)$	Inverse
$P(B\|A)$	Conditional
$P(B,A) = P(B\|A)\,P(A)$ $= P(A\|B)\,P(B)$	Product Rule
$P = (B,A) + (B,\overline{A})$ $= P(B\|A)\,P(A) + P(B\|\overline{A})\,P(\overline{A})$	Sum Rule



Doctor Bayes' test

- 99% of people with the disease test positive
- 1% of healthy people test positive
- Occurs randomly in 1 out of 10,000 people

What is the probability that you test positive?

$$P(disease|test\ =\ pos)\ =\ \frac{P(pos|disease)\cdot P(disease)}{P(pos)}$$

1. Likelihood $P(pos|disease)\ =\ 0.99$
2. Prior $P(disease)\ =\ 1/10\,000\ =\ 0.0001$
3. Marginal Probability
   $P(pos)\ =\ P(pos|disease)P(disease)\ +\ P(pos|no\ disease)P(no\ disease)$
   $=\ (0.99\ *\ 0.0001)\ +\ (0.01\ *\ 0.9999)$
   $=\ 0.010098$
4. Posterior
   $P(disease|pos)\ =\frac{0.99*0.0001}{0.010098}=\ 0.0098$

Generative Bayesian classifier

The Naive Bayes Classifier