AINT351 - Revision for Ian

Three types of learning

Imagine a machine experiences a sequence of sensory inputs `x1, x2, ... xn`

Supervised learning:

- The machine is also given `y1, y2, ..yn` and its goal is to learn and reproduce them from the inputs
- Learning by examples, input and output is given so it knows how to reproduce the output from the input

Unsupervised learning:

- The machine should build a representation of x that can be used for decision making, prediction
- There is no desired output, you are given inputs and after some iterations you start to categorise data based on some criteria

Reinforcement learning:

- The machine can generate actions `a1, a2,.. an` that affects its environment and receives a reward or punishment based on them. Its goal is to learn actions that maximise long term reward
- Learning based on rewards for actions so that it learns to maximise long term reward

Goals of supervised learning

Classify input data:

- In this case the desired outputs `y1,y2,...yn` are discrete class labels and the goal is to **classify** new output correctly from the new input
- have an image of a digit and want to know what digit it is based on previous examples of that digit

Goals of unsupervised learning

Regression

- In this case the desired outputs `y1, y2, ...yn` are continuous values and the goal is to **predict** new output correctly from new input
- Have the data from babies and can try to predict its weight given its height

We wish to find useful representations of data. This can involve

- Finding clusters
- Dimensionality reduction

- Finding the hidden cause of the surface phenomena
- Modelling the data probability density
- Data compression

Probability

Types of data:

Discrete data: only certain values

- Dice value = {1,2,3,4,5,6}
- Flip a coin = {H,T}

Continuous data: any value

- Length measurement
- Weight measurement

Probability functions

- A probability function maps possible values of a variable to its respective probabilities
  - e.g. if value is x we can write its possible probabilities as p(x)
- Probability functions have the following properties
  - P(x) is a number with a value between 0 to 1.0
  - The area under a probability function is always unity

The addition law of probability

- If two events A and B are mutually exclusive then
- `P(A U B)` = the probability event A **OR** B occurs
- `P(A U B) = P(A) + P(B)`
- If two events A and B are **NOT** mutually exclusive then
- `P(A U B) = P(A) + P(B) - P(A n B)`
- You have to subtract the intersect as it is where both events happen

Probability distributions

Bernoulli distribution:

- The probability of a success of failure, heads or tails, 1 or 0
- `n` is the number of times that the experiment is repeated

Discrete distribution:

- A finite amount of probabilities all of which have equal probability of occurring
- A dice throw, each outcome has a probability = 1/6

Cumulative probability:

- The probability of this event happening **ASWELL AS** all the previous events
- A dice landing on 6 as well as all the chances of it landing on 1,2,3,4 and 5 = 6/6

Binomial distribution:

- 2 outcomes
  - Heads or tails
- What is the probability of getting exactly 3 heads in 5 coin tosses
- HHHTT
  - `(1/2)^3 x (1/2)^2`
- THHHT
  - `(1/2)^1 x (1/2)^3 x (1/2)^1`
- All equal = `(1/2)^3 x (1/2)^2`
- therefore the overall probability =
  - `N x (1/2)^3 x (1/2)^2`
  - where N = number of unique arrangements
- There are exactly 10 ways to get 3 heads in 5 coin tosses
  - N = 10
- `10 x (1/2)^3 x (1/2)^2 = 0.3125`

Uniform distribution:

- A distribution that has constant probability
- 0.5 of values 0 and 1.0

Continuous data distributions:

- A continuous random variable is a random variable with a set of possible values that is infinite or uncountable
- looks like Gaussian distribution

Variance

- It measures how far a set of random numbers are spread out from their mean
- variance is the expectation of the squared deviation of a random variable from its mean

Expected value

- What is the expected value of `x` given it is in a certain area under a curve
- To find:
  - Get the marginal distribution by summing all the probabilities in that row or column (for that value of x or y)
  - Multiply the value of x by the marginal distribution of that value

Covariance: Joint probability

- The covariance measures the strength of the linear relationship between two variables
- to get the covariance of expected values:
  - find the expected value of x and expected value of y by using steps described above
  - get the expected value of xy = E(XY) by multiplying each value in the table by its x and y values
  - and then plug all values into equation
  - `COV(XY) = E(XY) - E(X) . E(Y)`

Coefficient of XY

- To get the coffecient of XY we need the standard devation of x and standard deviation of y as well as the covariance figured out in steps above
- ‘Coff(XY) = COV(XY) / STD(X) . STD(Y)
- First get the variance of x and y
  - `variance of x = E(X^2) - E(X)^2` - to get the `E(X^2)` need to do it the same as the expected values by square all the values of x
  - `variance of y = E(Y^2) - E(Y)^2` - to get the `E(Y^2)` need to do it the same as the expected values by square all the values of Y
- Once you have the variance you can square root it to get the standard deviation
- then plug it all back into the equation

Conditional probability distribution, independence

- How can you tell if x and y are independant
- Changing the value of y should have no effect on the probability distribution of x

Effect of standard deviation

- The variance either side of the mean

- The greater the standard deviation the greater the probability that x is within it
- Greater the standard deviation the further it is from the mean

Cumulative distribution function

- For a continuous random variable X the cumulative distribution function
- CDF(x) represents the area under the probability density function P(x) to the left of X
- `CFD(x) = P(X < x)`

Exponential distribution

- Given `P(x) = e^-x`
- What is the probability of x falling withing 1 to 2
- `P(1 <= x <= 2)` corresponds to area under distribution between 1 and 2

Marginalization

- sum up all the columns and row values for either x or y respectively

Conditional probability: bayes rule

- ‘P(A n B) is the probability of A and B happening
- can be written P(A, B) - the joint distribution of A and B
- The probability of A happening multiplied by the probability of B happening given that A has happened

$$P(A \cap B) = P(B|A)P(A)$$

$$P(B|A) = \frac{P(A \cap B)}{P(A)}$$

- if A and B are **NOT independent events** then - P(A n B) = P(A) . P(A|B)
- **This relationship is always true**

Interpreting covariance

- If we calculate the covariance between two random variables
- if the cov(X, Y) > 0
  - X and Y are positively correlated
- If the cov(X, Y) < 0
  - X and Y are inversely correlated
- If the cov(X, Y) $= 0$
  - X and Y are independent

Marginalisation

- Sum of probabilities across their given variable

Conditional probability

- If A and B are **NOT** independent events then
- `P(A n B) = P(A) . P(B|A)`
- This relationship is always true
- A has to happen for it to be true
- If A and B **ARE** independent events then
- P(A n B) = P(A) . P(B)
- This relationship is only true if A and B are independent
- Rolling a dice:
  - The first throw doesn't effect the second

Product rule of probability

- Product rule states that
- `P(A,B) = P(B)P(A|B)`
- `P(A,B) = P(A)P(B|A)`
  - The joint probability of A and B is prob of A multiplied by the probability of A given B
    - Same the other way around
- Leads to bayes rule
- `P(B)P(A|B) = P(A)P(B|A)`
- `P(A|B) = P(A)P(B|A)/P(B)`
  - The probability of A given B is equal to:
  - The probability of A multiplied by the probability of B given A

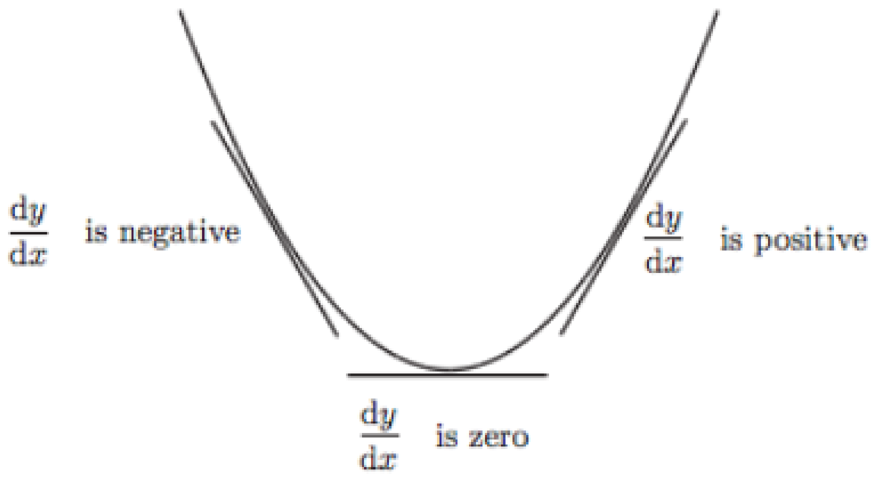
    - Over the probability of B
    - This is bayes theorem
- If A and B are independent
    - `P(A, B) = P(A)P(B)`
    - Joint probability of two independent events is the dot product probability of both events happening

Gradient descent

- How can we get to the top of a hill?
    - We follow the gradient
- How can we get to the bottom of a hill
    - We follow the descent
- How do we know we're at the top
    - It goes down on both sides

Gradient of a curve

- Gradient is the slope of a curve or surface
- Going up the hill it is +ve
- Going downhill it is -ve
- Differentiation finds tagent of line on graph
- Gradient of straight line =
    - Change in y
    - over change in x



Iterative gradient decent

- Find minimum of a function

- Move downwards in direction of gradient

Local maxima and minima

- Can be local minima as well as global minima
  - Same for maxima
- Gradient descent can get stuck in local minima/maxima

Least squares fitting

- We want to fit a straight line to data measurements
- Equation for straight line in a single dimension is:
  - yi = mxi + c
- Sum error over all points

Generate a N dimension Gaussian distribution

- If we use `randn(N, 1)` to draw N samples from a 1D distribution x1
- And use it again to draw N samples from a 1D distribtion x2
- From these we build a 2D vector X
- What is the covariance matrix of 2D dataset X
  - E = [1 0; 0 1]
- Covariance terms = 0 and 0
- Means that they are independant

Terminology for types of learning

Maximum likelhood (MP) learning:

- Does not assume a prior over the model parameters
- Finds q parameter settings that maximises the likelihood of the data
- ML may run into estimation problems if we're unlucky

Maximum a posteriori (MAP) learning:

- Assumes a prior over the model parameters
- Finds a parameter settings that maximises the posterior
- Having a prior belief means that the intial estimate may be more appropriate

Bayesian learning:

- Assumes a prior over the model parameters
- Computes the posterior of the parameters

NB: MAP and ML estimates are identical when the prior is uniformly distributed

Gaussian class-conditional model

- The maximum likelihood fit of a Gaussian to some data is the Gaussian whose mean is equal to the data mean and whose covariance is equal to the sample covariance
- One nice feature of this model is that the MP parameters can be found in closed-form:
  - So we don't have to use iterative solutions

Limitations:

- We cannot account for higher order statistical structure in the data
  - These require nonlinear and hierarchical models
- We need to deal with outliers
  - These require nonlinear and hierarchical models
- The multivariate model uses D(D+1)/2 parameters
  - If D is very large we need to use dimensionality reduction

Frequentest and Bayesian statistics

Frequentest approach:

- Probability is the limit of observed frequency as number of observations goes to infinity
- Considers the model parameters to be fixed (but unknown) and calculates the probability of the data given those parameters

Bayesian approach:

- Probability is a "degree of confidence" that one attaches to an uncertain event
- Requires a prior estimation of the models likelihood
- naturally incorporating prior knowledge

Eigenvalues and Eigenvectors

- If A is an nxn matrix
- A scalar lambda is called an eigenvalue of A
  - If there is a non-zero vector x such that
  - Ax = lambdax
- If we have a matrix and a scalar value
  - A and lambda
- If both of them multiplied the vector x
  - result in the same vector values of x

- Then such a vector x is called an eigenvector of A corresponding to lambda
- Eigenvectors can only be found for square matrices
- Not every square matrix has an eigenvector
- An `n x n` matrix can have n eigenvectors
- All the eigenvectors of a matrix are orthogonal
- The length of an eigenvector is exactly one?
- When we transform X by multiplying by A we end up with vector X again but this times scaled by lambda
  - Therefore the **direction of vector X is unaffected by the transformation**
- Only keeps the vectors of parallel values

Clustering

- Idea of clustering is to group patterns so that:
  - Patterns that are similar to each other are in the same cluster
  - Patterns that are dissimilar to those are in the other clusters
- All require a way to determine similarity

K-means clustering

- K is the number of clusters **Steps:**
- Assign K
- Randomly assign the first K data points to be the centroids of the K clusters
- While loop
  - For each point, assign it to its closest cluster centre
  - Re-compute the cluster centres as the means of assigned data points
  - Terminate loop if there was change in assignment compared to last iteration (converges)
- K-means is an example of hard clustering
  - Clusters do not overlap
  - Data in one cluster only

Mixture of Gaussians

- Gaussian, FA and PCA models are easy to understand and use in practice
- They are a convenient way to reduce dimensionality of high dimensional data sets
- The problem is that they make very strong assumptions about the distribution of the data
  - Only the mean and variance of the data are taken into account
- The class of densities which can modelled is also too restrictive
- By using mixtures of simple distributions
  - Such as Gaussians
  - We can expand the class of densities greatly
- Mixture of Gaussians is an example of soft clustering
  - Clusters may overlap
  - Data may exhibit non-binary strength of association to all clusters
- Mixture of Gaussians
  - Probabilistic method
  - Each cluster is a generative model
  - Clusters have parameters (mean & covariance)
  - Train using EM
- Advantages is that given enough components we can model most distributions using only the simple Gaussian distribution

EM Algorithm

- Need to know the Gaussian parameters (mean and standard deviation) for each cluster to estimate the data point cluster membership
- But need to know data assignments to estimate the parameters
- Solution is to use the EM algorithm
  - Randomly initialise the Gaussian cluster parameters
- E step:
  - Look at each data point and calculate how likely it came from a given cluster
  - Do so by computing p(b|xi)
- M step:
  - Re-estimate Gaussian parameters to fit the assigned points
  - Repeat until convergence

Pattern classification

- Pattern classifiers partitions the input space

- May have multiple input data dimensions
- May have multiple output classes
- Type of decision boundary depends on the classifier
- Variety of ways to determine boundaries

Naive Bayes classifiers

- Estimation of full high demsionality covariance matrix can be a problem
  - Why?

Problem:

- We can only estimate P(x1,x2,. . . ,xn|y) based on counts in the training dataset
- We may not see every x value for every y value

Solution:

- Make an assumption of independence between the features
- Assume conditional independence given y
- so for a dataset:
  - Calculate probabilities of each feature xi independently
  - Then joint probability is just the product of individual feature probabilities

Conditional independence

Example - Going to the beach and getting heatstroke not independent - `P(B, S) > P(B) . P(S)` - However they may be independent if we know that the weather is hot - `P(B, S | H) = P(B|H) . P(S|H)` - In this case the hot weather explains a reason for dependence between going to the beach and getting heatstroke - Similarly in classification the class values explains the dependencies between the attributes

Information theory

- The probability P(X) encodes uncertainty about the random variable X

Entropy

- We can quantify the information represented in such a random variable
- This is the entropy of the variable X which is average amount of information required to encode X

Mutual information

- It indicates how much knowledge if Y affects our belief in X
- If Y has no effect and is independent of X then the mutual information is zero

Naive Bayes versus full Gaussian classifier

- The probability of a data point
  - how probable is xi under source c
- Naive bayes only estimates and uses marginal Gaussian parameters
- Only has non-zero values in the covariance matrix along its leading diagonal
- Assumes variables are independent
  - zero covariance
- Can only represent distributions aligned with coordinate system
- Easy to estimate parameters
- Choose most probable class given observation

Limitations of generative models

- Bayes' decision rule minimise average probability of error
- Can train generative models by directly estimating parameters
- So what needs improving
- Bayes classifier is the minimum error classifier only if our model of the data is appropriate
- In particular if the form of the class conditional distribution is correct

Discriminative models

- Generate model classification requires the class posterior
- Discriminative models use parameters more closely related to classification process
- They are not dependent on generative process being correct
- Don't model the data just try to find decision boundary
- Example is the perceptron

Linear decision boundaries

- Decision boundaries partition the input space into discrete regions
- Each region is associated with a class label

The preceptron

- Learning means changing the weights between the neurons
- Relationship between input and output is important in computational neuroscience
- Simple but limited capabilities
- Basic concepts are useful for multi-layer models
- If the data is linearly separable then:
  - A linear decision boundary will correctly classify all points
  - Algorithm will stop where there are no incorrectly classified training data points
  - The algorithm therefore guaranteed to converge to a solution
- Two forms of update are generally adopted
- Batch update:
  - All the data is presented for each iteration
  - Weights only updated after all data seen
- Sequential update
  - Data is presented one sample at a time
  - Weights updated after each sample

What is deep learning?

- *'Deep learning'* involves using a neural network with several layers of nodes between the input and output
- A series of layers between the input and output do feature detection and processing in a series of stages
- Model of human visual system
- Use recent algorithms for training many-layer networks

Deep networks

- Two main types of deep networks:
- Convolutional neural networks
  - Employ alternating layers of convolutional networks followed by a pooling layer
  - Output uses traditional MLP
- Deep belief networks
  - Consist of perceptron stacked Blotzmann machinces
  - Use classification output layer

Auto encoder

- An auto-encoder is trained with standard training algorithm

- It learns to map the input back onto itself

New way to train multi-layer NNs

- Use greedy layer-wise training to train multilayer networks
- Train first layer using data unsupervised without labels
- Use abundant unlabeled data which is not part of the training set
- Freeze the first layer parameters and start training the second layer using the output of the first layer as the unsupervised input to the second layer
- Repeat this for as many layers as desired
- this builds a set of robust features
- Use the outputs of the final layer as inputs to a supervised layer/model and train the last syperviseed layers
    - leave early weights frozen
- Each of the non-output layers is trained to be an auto-encoder
- Essentially it is forced to learn good features that describe what comes from the previous layer

Advantages of greedy layer-wise training

- Avoids many of the problems of trying to train a deep net in a supervised fashion
- Each layer gets full learning focus in its turn since it is the only current 'top' layer
- Can take advantage of unlabelled data
- When you finally tune the entire network with supervised training the network weights have already been adjusted so that you are in a good error basin and just need fine tuning
- This helps with problems of
    - Ineffective early layer learning
    - Deep network local minima

Convolution networks

The replicated feature approach

- Use many different copies of the same feature detector with different positions
    - Could also replicate across scale and orientation
    - Replication greatly reduces the number of free parameters to be learned
- Use several different feature types, each with its own map of replicated detectors
    - Allows each patch of image to represented in several ways

Effect of replicating feature detectors

- **Equivariant activities:** Replicated features do **not** make the neural activities invariant to translation
  - The activities are equivariant
- **Invariant knownledge:** If a feature is useful in some locations during training
  - detectors for that feature will be available in all locations during testing

Pooling replicated feature detectors outputs

- Get a small amount of translational invariance at each level by averaging four neighbouring replicated detectors to give a single output to the next level
- This reduces the number of inputs to the next layer of feature extraction
  - Thus allowing us to have many more different feature maps
- Taking the maximum of the four works slightly better
- **Problem**
  - After several levels of pooling we have lost information about the precise positions of things
  - This makes it impossible to use the precise spatial relationships between high-level parts for recognition

Convolutional networks

- Convolution is the integral of the product of the two functions after one is reversed and shifted
- Same as sliding a single instance of the detector over input
- Performs convolution like an FIR filter
- Each convolution layer contains multiple feature maps
  - Just as hidden layers typically multiple nodes
  - So each map is a single feature detector
- Repeatedly applied to a small window across the whole input
- A convolutional neural network is tiled in such a way that they respond to overlapping regions in the input field
- Convolutional layer consists of unites that act as feature detectors
- Each convolutional unit is connected to a small region of the input
- This structure is replicated for all offsets across the input of data to span there input space

- Thus the augmented connection weights are shared for all the hidden units in this feature map
- Each layer combines patches from previous layers
- Typically tries to compress large data into smaller set of robust features based on local variations
- Convolution can create many features

Pooling layer

- A pooling layer often follows a convolution layer
- It down samples the feature map
- This step can compress and smooth the data
- Make data invariant to small translational changes
- Usually takes the average or max value across disjoint patches
- Feature outputs are pooled independently

Final MLP layer

- To make decisions a final MLP is often used
- This can then make discrete classifications of the input

The perceptron

- Bias can be augmented into weight vector
- Also add corresponding unity value in augmented data vector

The perceptron learning rule

- TO train the perceptron
- first initially set all weights to small random values
- Then loop
  - For all data point and its output target in turn, compute perceptron output z (either 1 or a 0)
  - After each data point presentation the weight update is given
  - Weight of input i perceptron node
  - multiplied by the target + 1 is equal to
  - this weights input i perceptron node
  - multiplied by the target
  - plus the learning rate multiplied by target for current input
  - minus the current output
  - times x which is the ith input

- Need to repeat procedure across all the dataset until finished
- Guaranteed to converge if the problem is separable

Perceptron rule versus delta rule

- Perceptron provides learning feedback after a threshold non-linearity
- Instead can provide feedback before the threshold
- Replace step function with continuous differentiable function
- Can be linear - if so, problem similar that of linear regression
- can still use threshold for final classification
- This has the advantage we can use gradient descent to find weights

Single vs multi layer networks

- Single layer networks implements linear decision boundary
- Multilayer networks can implement more complex decision boundaries

Kernel trick

- Transforming some datasets can make it linear separable
- more rigorously this can be done using a kernel
- A kernel is basically a mapping function that transforms one given space into another
- Such transformation of data that leads to easier to solve problems is sometimes known as the kernel trick

Limitation of back propagation

- Multiple hidden layers
- get stuck in local optima
  - Start weights from random positions
- Slow convergence to optimum
  - Large training set needed
- Only use labelled data
  - Most data is unlabelled
- Error attenuation with deep nets
- Can now training with GPUS and special hardware
- However can still be time intensive for large networks