# IAML Notes

## Lecture 1

- Supervised Learning
  - Predict output y when given input x
  - Categorical y: classification
  - Real-valued y: regression
- Unsupervised Learning
  - Internal representation of the input e.g. clustering, dimensionality
  - Getting labels is difficult and expensive
- Other areas of ML
  - Learning to predict structured objects e.g. trees, graphs
  - Reinforcement learning - learning from "rewards"
  - Semi-supervised learning - combination

#### Classification

**No text !!!**

#### Regression

$$f(x) = w_0 + w_1 x_1 + w_D x_D$$

$$x = (x_1, \ldots, x_D)^T$$

- $f(x)$ is a linear function
- Setting of $w_1, \ldots, w_D$ is done by minimising the cost function
- Usual score function is $\sum_{i=1}^{n} = (y^i - f(x^i))^2$.

#### Clustering

**No text !!!**

#### Structure of Supervised algorithms

- Define the task
- Decide on the model structure (choice of inductive bias)
- Decide on the score function (judge quality of fitted model)
- Decide on optimization/search method to optimize the score function

Supervised learning is inductive, i.e. we make generalizations about the form of $f(x)$ based on instances D.

Learning is impossible without making assumptions about f!!

**Conditional Probability**

$$p(X = x|Y = y) = \frac{p(x, y)}{p(y)}$$
$$p(X, Y) = p(Y)p(X|Y) = p(X)p(Y|X)$$

From the product rule,

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}$$

From the sum rule the denominator is

$$p(X) = \sum_y p(X|Y)p(Y)$$

Say that $Y$ denotes a class label, and $X$ an observation. Then $p(Y)$ is the prior distribution for a label, and $p(Y|X)$ is the posterior distribution for $Y$ given a data point $x$.

**Conditional independence**

$$p(X1|X2, Y) = p(X1|Y)$$

**Marginal independence**

$$p(X|Y) = P(X)$$

**Continuous Random Variables**

$$\mu = \int xp(x)dx$$
$$\sigma^2 = \int (x - \mu)^2 p(x)dx$$

For numerical discrete variables, convert integrals to sums.

**Gaussian distribution**

**1D**

$$p(x|\mu, \sigma) = N(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left\{-\frac{(x - \mu)^2}{2\sigma^2}\right\}$$

- $\sqrt{2\pi\sigma^2}$ is the normalisation constant

**More D**

$$p(x) = \frac{1}{(2\pi)^{d/2}|\mathbf{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(x - \mu)^T \mathbf{\Sigma}^{-1}(x - \mu)\right\}$$

- $\Sigma$ is the covariance matrix

$$\Sigma = E[(x - \mu)(x - \mu)^T]$$
$$\Sigma_{ij} = E[(x_i - \mu_i)(x_j - \mu_j)]$$

### Likelihood

The probability of data D given model M is $p(D|M)$. This is called the **likelihood**.

$$p(D|M) = \Pi_{i=1}^{N} p(x_i|M)$$

i.e. the product of generating each data point individually.

This is a result of the independence assumption. Try different models, pick the one with highest likelihood -> **Maximum likelihood approach**.

### Bernoulli distribution

To find most probably number of ones in $n$ tosses, differentiate with respect to p, to find when it is maximised. Useful to take a log - it is monotonic.

$$\prod_i p(x_i|\mu, \sigma^2) = -\frac{1}{2}\sum_i \frac{(x_i - \mu)^2}{\sigma^2} - \frac{n}{2} log(2\pi\sigma^2)$$

$$\mu = \frac{\sum_i x_i}{n}$$

$$\mu = \frac{\sum_i (x_i - \mu)^2}{n}$$

## Lecture 2

## Attribute-value pairs

- unordered bag of features, if structure essential, embed it
- categorical - red, yellow, blue
  - mutually exclusive
  - only equality meaningful
- ordinal - bad, good, better, best
  - there is natural ordering
  - comparison is meaningful, maths isn't
- numerical - 1, 3, 8, 4
  - usually need to normalise, remove outliers
  - maths is meaningful

### Skewed distributions

- Affects regression, kNN, NB; but not DTs
- fix: take a log or atan, then normalise
- cumulative distribution function $x' = F(x) = P(X < x)$

**Non-monotonic effect of attributes**

- age -> probability of boxing win (not too young, not too old)
- affects regression NB, DTs (gain); less important for kNN
- if there's an obvious way to make monotonic: use it

**Picking attributes** - instances in the same class y should have representations x that are somehow "similar".

**Object recognition** - segment image into regions, compute features describing the region.

**Dealing with structure** - Represent path from root to leaf as a feature.

**Outliers** - Isolated instances of a class that are unlike any other instances observed. Remove them or threshold.

**Missing values** - Try to understand why, if categorical have special category, if numerical then fill in mean or remove them. Some learners handle it.

## Lecture 3

**Bayesian classification**

$$P(y|x) = \frac{P(x|y)P(y)}{\sum_{y'} P(x|y')P(y')}$$

- P(y): prior probability of each class
  - encodes how which classes are common, which are rare
  - apriori much more likely to have common cold than Avian flu
- P(x|y): class-conditional model
  - describes how likely to see observation x for class y
  - assuming it's Avian flu, do the symptoms look plausible?
- P(x): normalize probabilities across observations
  - does not affect which class is most likely (arg max)

An outlier has a low probability under every class.

Independence assumption - assume $x_1, x_2 \ldots$ conditionally independent given $y$.

$$P(x_1 \ldots x_d|y) = \prod_{i=1}^{d} P(x_i|x_1 \ldots x_{i-1}, y) = \prod_{i=1}^{d} P(x_i|y)$$

Probabilities of going to the beach and getting a heat stroke are not independent. May be independent if we know the weather is hot. Hot weather *explains* the dependence between beach and heatstroke. Thus, **class value explains all the dependence between attributes**.

**Decision boundary**

- Different means, same variance: straight line / plane
- Same mean, different variance: circle / ellipse
- General case: parabolic curve

**Problems with Naive Bayes**

- Zero frequency problem: add-one (Laplace) smoothing

- Word independence: add lots of hammy words to spam email
- Unable to handle correlated data - double weight to "same" features

**Good stuff about Naive Bayes**

- Missing data: leave out attribute that's missing (independence assumption).
- Good complexity
  - $O(nd + cd)$ training time complexity
  - $O(dc)$ space complexity
  - $O(d)$ insertion / deletion - store partial sums instead of mean/variance
  - c - classes, n - instances, d - attributes

**Naive Bayes structure**

- Task: c-class classification
- Model: $c \times d$ independent assumptions
  - Continuous: Gaussian, Discrete: Bernoulli
- Score function: class-conditional likelihood
- Optimisation: analytic solution

## Lecture 4

Decision trees - try to *understand* when John plays tennis. Split into sets, are they pure? If not repeat, if yes then done and see which subset new data falls into.

**ID3 Algorithm**

1. A <- the best attribute for splitting the examples
2. Decision attribute for this node <- A
3. For each value of A, create new child node
4. Split training examples to child nodes
5. If examples perfectly classified then stop else iterate over new child nodes.

**Which attribute to split on?**

Want to measure purity of the split so we are more certain after the split. We use entropy:

$$H(S) = -p_{(+)}log_2p_{(+)} - p_{(-)}log_2p_{(-)}$$

- S: subset of training examples
- $p_{(+)}/p_{(-)}$: % of positive/negative examples in S

**Information Gain**

We want many items in pure sets and expect drop in entropy after split:

$$Gain(S, A) = H(S) - \sum_{V \in Values(A)} \frac{|S_V|}{|S|} H(S_V)$$

**Overfitting**

We can always classify training examples perfectly - keep splitting until each node has 1 example. Doesn't work on new data.

To avoid this, we stop splitting when not statistically significant. Grow the tree, then post-prune based on a validation set.

**Sub-tree replacement pruning (WF 6.1)**

- for each node
  - pretend to remove node + all children
  - measure performance on validation set
- remove node that results in greatest improvement
- repeat until harmful

**Structure of DT**

- Task: classification, discriminative
- Model: decision tree
- Score function: information gain at each node, prefer short trees and high-gain near root
- Optimisation: greedy search from simple to complex guided by information gain

**Problems of DT**

- Biased towards attributes with many values
- Doesn't work on new (un-observed) data
- Only axis aligned splits of data

**GainRatio**

Idea is to penalise attributes with many values.

$$SplitEntropy(S, A) = -\sum_{V \in Values(A)} \frac{|S_V|}{|S|} log \frac{|S_V|}{|S|}$$

$$GainRatio(S, A) = \frac{Gain(S, A)}{SplitEntropy(S, A)}$$

Continuous DTs - Create ranges to split on which can be optimised (WF 6.1).

**Multi-class classification**

- predict most frequent class
- Entropy: $H(S) = -\sum_c p_{(c)} log_2 p_{(c)}$
- $p_{(c)}$ % of examples of class c in S

**Regression**

- predicted output: mean of the training examples in subset
- requires a different definition of entropy (dunno)
- can use linear regression at the leaves (WF 6.5)

**Good stuff about DTs**

- interpretable
- easily handles irrelevant attributes (Gain = 0)
- can handle missing data - (WF 6.1)
- very compact # of nodes << d after pruning
- very fast at testing time $O(depth)$

**Random decision forest**

- Grow K different decision trees
  - Pick a random subset S_r of training examples
  - grow a full ID3 tree T_r (no pruning)
    - when splitting pick from d << D random attributes
    - compute gain based on S_r instead of full set
  - repeat for r = 1 ... K
- Given new data point X
  - classify X using each tree and then use majority vote (state-of-the-art performance)