CS3237

Lecture 2

Statistical Methods

colintan@nus.edu.sg

National University of Singapore

School of Computing

Introduction

- **Statistical methods, as the name implies, uses statistical formulae to learn from data and make predictions.**
- **We will look at four major methods:**
  - Regression.
  - Naïve Bayes Classification.
  - Decision Trees
  - Support Vector Machines.
- **This lecture gives you an alternative to neural networks and deep learning.**
  - Statistical methods have better theoretical basis.
  - Statistical methods in general are faster to build and train.
  - Statistical methods usually have fewer hyper-parameters to adjust.
- **Statistical methods also give you a good idea whether the data is actually "learnable"**
  - Much faster prototyping time, much simpler, though may provide relatively poor results.

SWS3009 Embedded Systems and Deep Learning

REGRESSION : LINEAR MODELS

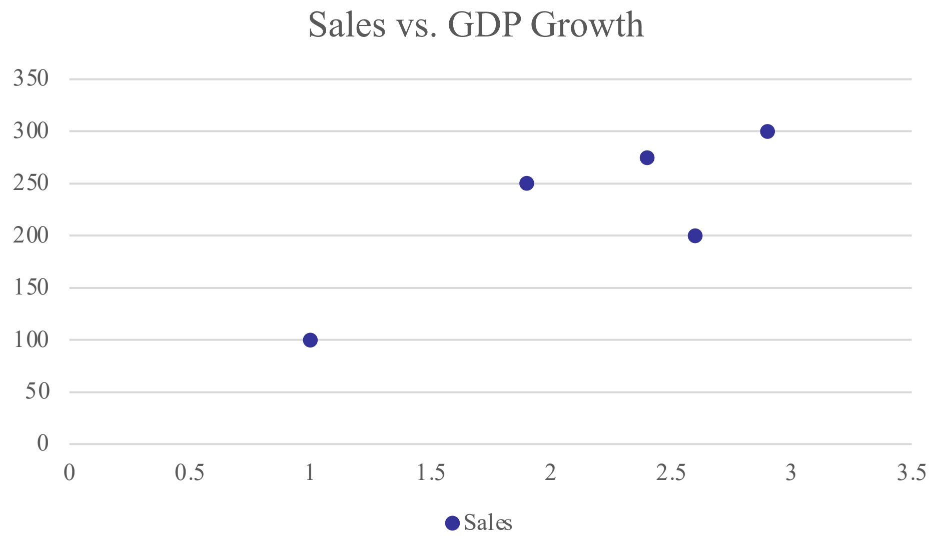
Statistical – Regression

- **In regression analysis:**
  - We have data that shows the relationship between a dependent variable and one or more independent variables.
    - ✓**E.g. the relationship between population growth and time.**
  - We want to estimate this relationship.
- **The relationship between the dependent variable and the independent variable can be a straight line (linear) or not (nonlinear).**
  - We will only consider linear relationships with one independent variable (simple linear regression).
  - In the hands-on we will look at multivariable linear regression.
  - We skip non-linear regression as this is served well by neural networks and deep learning techniques that we have already seen.

Simple Linear Regression

- **Let's suppose your company has data about sales figures versus GDP growth:**
  - ▪Independent variable: This is the variable that you cannot control. Here it is GDP growth.
  - ▪Dependent variable: This is the variable you are interested in predicting, against the independent variable. Here it is sales.

Change in GDP	Sales
1	100
1.9	250
2.4	275
2.6	200
2.9	300



Simple Linear Regression Assumptions

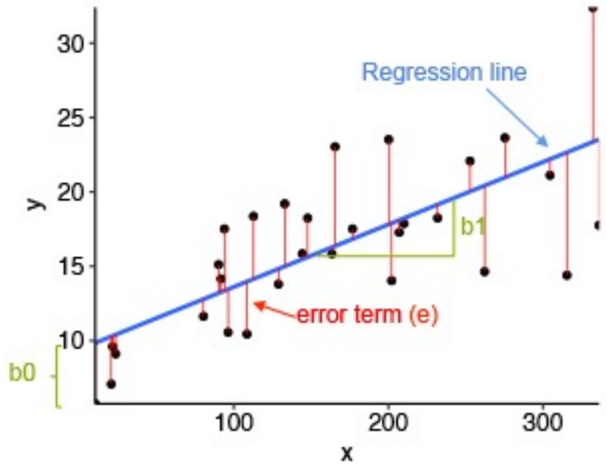
- **In linear regression we assume:**
  - There is correlation between the dependent (y) and independent (x) variables.
  - y depends on x through a linear equation:

$$y_i = ax_i + b$$

  - However there is random noise *e* in the observations of the dependent variable:

$$y_i = ax_i + b + e_i$$

  - Our task is thus to find *a* and *b*. BUT FIRST we must test if there is indeed a relationship between *x* and *y*.



Simple Linear Regression
Covariance

- **Our plot appears to show that sales figures change when GDP figures change. Hence there seems to be a “covariance”.**

$$\boldsymbol{cov(x,y) = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{n-1}}$$

- **Hence we first calculate the averages for the GDP and Sales figures:**
  - Average(GDP) = (1+1.9+2.4+2.6+2.9)/5 = 2.16
  - Average(Sales) = (100+250+275+200+300)/5 = 225
  - Cov(Sales, GDP) =

    ((1-2.16)(100-225)+(1.9-2.16)(250-225)+(2.4-2.16)(275-225)+(2.6-2.16)(200-225)+(2.9-2.16)(300-225))/4

    = 48.75
- **Since this number is positive, sales figures grow in the same direction as GDP.**

Simple Linear Regression Correlation

- **The covariance just tells us the direction in which the dependent variable grows according to the independent variable, and the relative strength. We need to scale to between -1 and 1 in order to see how well correlated these two variables are.**
  - The standard deviation of a variable x is given by:

$$s_x = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

The correlation between *x* and *y* is given by:

$$\rho = \frac{cov(x,y)}{s_x s_y}$$

Simple Linear Regression
Correlation

- **Given our sample data, $s_x = 0.74$ and $s_y = 79.06$, giving is a correlation of:**

$$\boldsymbol{\rho = \frac{48.75}{(0.74)(79.06)} = 0.83}$$

- **Since this number is quite close to 1, we see that sales figures are strongly correlated to GDP growth.**
  - Hence it makes sense for us to try to create a linear model for the two variables.

Simple Linear Regression Parameter Estimation

- **Recall that we assume that our dependent variable *y* depends on *x* through the following relationship:**

$$y_i = ax_i + b + e_i$$

- **The error is therefore:**

$$\boldsymbol{e}_i = y_i - ax_i - b$$

- **We want to derive *a* and *b* that minimize this error *e* over all sample data points *i*. This is equivalent to minimizing Q(a,b), where:**

$$\boldsymbol{Q}(\boldsymbol{a},\boldsymbol{b}) = \sum_{i=1}^{n} \boldsymbol{e}_i^2$$

$$= \sum_{i=1}^{n} (y_i - ax_i - b)^2$$

Simple Linear Regression Parameter Estimation

- **To minimize Q(a,b), we take:**

$$\frac{dQ(a,b)}{da} = 0$$

$$\frac{dQ(a,b)}{db} = 0$$

- **We do the second equation first because it is simpler:**

$$\frac{dQ(a,b)}{db} = -2\sum_{i=1}^{n}(y_i - ax_i - b)$$

- **Set this to 0 and solving for b we have:**

$$b = \overline{y} - a\overline{x}$$

- **This means that the y intercept of our line is dependent on the averages of both x and y. Makes sense because our line should pass through the centre of the point cloud.**

Simple Linear Regression Parameter Estimation

- **Now we do the same for the first equation:**

$$\frac{dQ(a,b)}{da} = -2\sum_{i=1}^{n} x_i(y_i - ax_i - b)$$

$$0 = \sum_{i=1}^{n} (x_i y_i - a x_i^2 - b x_i)$$

$$0 = \sum_{i=1}^{n} (x_i y_i - a x_i^2 - (\bar{y} - a\bar{x}) x_i)$$

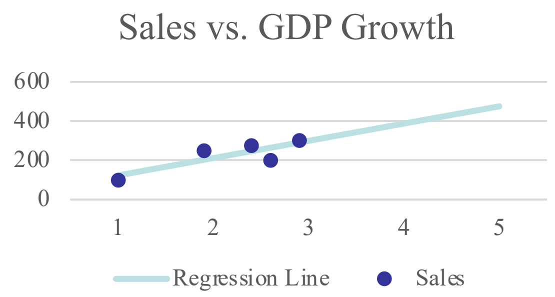
- **After a bit of algebraic black-magic, we solve for a and get:**

$$a = \frac{cov(x,y)}{s_x^2}$$

Simple Linear Regression Example

- **We are now able to find the regression line for our data:**

Change in GDP	Sales
1	100
1.9	250
2.4	275
2.6	200
2.9	300



- **Average GDP Growth = 2.16, Average Sales = 225**
- **Cov(GDP, Sales) = 48.75, Var(GDP)=0.553**
- **a = 48.75 / 0.553 = 88.16**
- **b = 225 – 88.16 * 2.16 = 34.57**
- **So our regression line is y = 88.16x + 34.57**
- **If GDP growth this year is 2.5%, we can now predict sales of (88.16 * 2.5 + 34.57) = 255.1 units.**

Linear Regression Hands-on

Start up Jupyter and load the "Linear Regression.ipynb" notebook.

1. **Change to your module directory you created in Lecture 1.**
2. **Copy over "Linear Regression.ipynb" to that directory.**
3. **Activate the virtual environment:** `source venv/bin/activate`
4. **Run Jupyter:** `jupyter notebook`

SWS3009 Embedded Systems and Deep Learning

CLASSIFICATION: NAÏVE BAYES METHOD

Naïve Bayes Classifier

- **While linear regression is good for predicting the value of a dependent variable given an unknown independent variable, they cannot conveniently used for classification problems:**
  - Classification Problem:
    - ✓**Given a set of data X, and a set of classes C, we want to compute the likelihood of some $x \in X$ belongs to some class $c_k \in C$.**
    - ✓**In particular, we want to find some c_{max} such that $P(c_{max} \mid x) = max_{ck \in C}\ P(c_k \mid x)$, then we can claim that x belongs to class c_{max}.**
  - E.g. problem:
    - ✓**X is a set of news articles, and x is one article.**
    - ✓**C is a set of news article types, e.g. sports, politics, home news, crime, etc.**
    - ✓**Our task then is to decide what kind of article x is.**

Naïve Bayes Classifier Assumptions

- **Assume that *x* is a vector of "features" *($x_1, x_2, \ldots, x_n$)*. Then we have for each $c_k \in C$ we want to compute $P(c_k \mid x_1, x_2, \ldots, x_n)$. In our example, *x* might consist of a set of words in an article, with each x_i being a single words.**
- **Using Bayes Law:**

$$p(c_k|x) = \frac{p(c_k)p(x|c_k)}{p(x)}$$

- **Since the denominator does not involve c_k and we are interested only in the relative ranking of each $c_k \in C$, we eliminate the denominator giving:**

$$p(c_k|x) = p(c_k)p(x|c_k)$$

- **This is equivalent to the joint probability model $p(x_1, x_2, \ldots, x_n, c_k)$**

Naïve Bayes Classifier Assumptions

- **Using the chain-rule:**

$$p(c_k, x_1, x_2, \ldots, x_n) = p(x_1 | x_2, x_3, \ldots, c_k)\, p(x_2 | x_3, \ldots, c_k) \ldots p(x_n | c_k)\, p(c_k)$$

- **In practice the joint probabilities $p(x_1|x_2,\ldots,c_k)$ etc. are difficult to estimate, and thus we make the naïve assumption that all x_i terms are independent. Thus each $p(x_i|x_{i+1}, \ldots, x_n, c_k) = p(x_i|c_k)$**
- **This gives us the final form for the Naïve Bayes Classifier**

$$p(c_k | x_1, x_2, \ldots, x_n) = p(c_k) \prod_{i=1}^{n} p(x_i | c_k)$$

Naïve Bayes Classifier Parameter Estimation

- **Naïve Bayes Classifiers must be "trained" on training data before being able to classify a given sample.**
- **Parameter Estimation:**
  - We need to estimate $p(c_k)$ for every $c_k \in C$:
    - ✓**Uniform Assumption: For *n* classes, the probability of $p(c_k)$ is simply 1/n.**
    - ✓ **Based on training set: if there is a total of *S* training samples, and**
    - ✓ **s_k samples for class c_k, then $p(c_k) = s_k / S$.**
  - Estimating $p(x_i|c_k)$ is somewhat trickier. We will look at three approaches:
    - ✓**Gaussian: Assumes that the probability of a feature x_i taking a particular value v in class c_k follows a normal distribution.**
    - ✓**Multinomial: Assumes that the frequency of a feature x_i occurring in a class is governed by a multinomial distribution.**
    - ✓**Bernoulli: Assumes that the boolean (yes/no) occurrence of a feature x_i occurring in class c_k is governed by a Bernoulli distribution.**

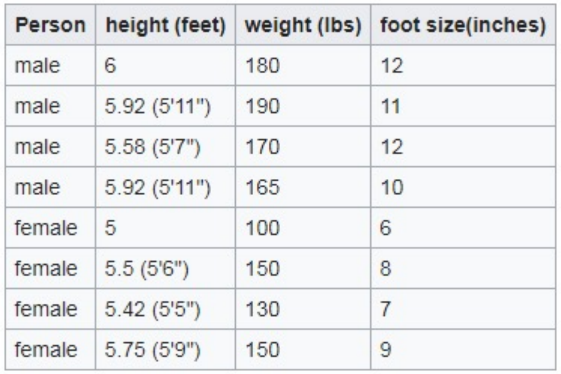
Normal Distribution Naïve Bayes Continuous Variables

- **This is used when classes contain continuous variables.**
  - E.g. when we are classifying male and female, we may consider height, shoe sizes, weight, etc. All of which are continuous values.
- **To compute $p(x_i|c_k)$:**
  - Partition all x_i by class c_k:
  - Find the mean μ_k and variance σ_k^2 of all data in c_k.
  - Now when we encounter a variable (e.g. height) x_i of value v (e.g. 1.8m), the probability density is:

$$p(x_i = v|c_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}}$$

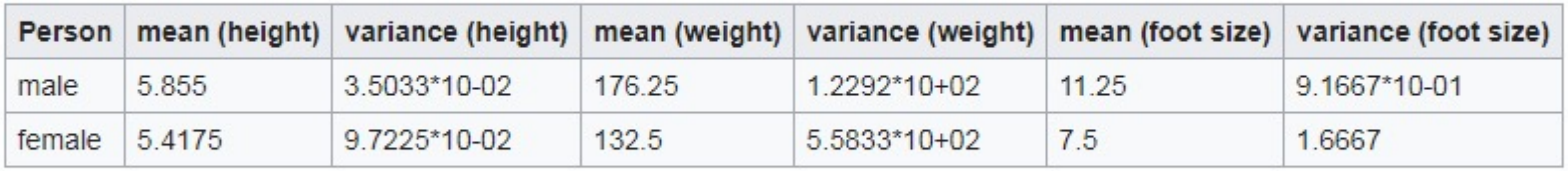
Normal Distribution Naïve Bayes Continuous Variables

- **Let’s take an example of classifying men and women based on height, weight and foot size (“borrowed” from Wikipedia):**



Person	height (feet)	weight (lbs)	foot size(inches)
male	6	180	12
male	5.92 (5'11")	190	11
male	5.58 (5'7")	170	12
male	5.92 (5'11")	165	10
female	5	100	6
female	5.5 (5'6")	150	8
female	5.42 (5'5")	130	7
female	5.75 (5'9")	150	9

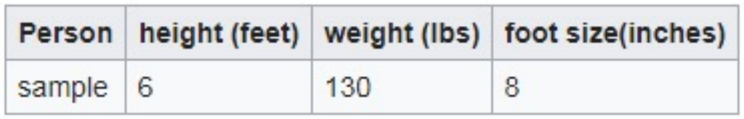
- **From here we need to work out the mean and variance for each class to compute the Gaussian probability function:**



Person	mean (height)	variance (height)	mean (weight)	variance (weight)	mean (foot size)	variance (foot size)
male	5.855	3.5033*10-02	176.25	1.2292*10+02	11.25	9.1667*10-01
female	5.4175	9.7225*10-02	132.5	5.5833*10+02	7.5	1.6667

Normal Distribution Naïve Bayes Continuous Variables

- **Assuming we have the following person and want to know if this is a male or female:**



Person	height (feet)	weight (lbs)	foot size(inches)
sample	6	130	8

- **Recall:**

$$p(c_k|x_1, x_2, \ldots, x_n) = p(c_k)\prod_{i=1}^{n} p(x_i|c_k)$$

  - Here it for the male class:

    p(male|height, weight, fs) = p(male)p(height|male)p(weight|male)p(fs|male)

  - Since our training set has 4 males and 4 females, p(male) = 0.5

Normal Distribution Naïve Bayes Continuous Variables

- **We compute p(height|male). This person is 6 feet tall, so we want:**

  p(height=6 | male):

$$\boldsymbol{p(height = 6|male) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(6-\mu_k)^2}{2\sigma_k^2}}}$$

$$\boldsymbol{p(height = 6|male) = \frac{1}{\sqrt{2\pi(3.5033\times10^{-2})}} e^{-\frac{(6-5.855)^2}{2\times3.5033\times10^{-2}}}}$$

=1.5789

- **We similarly find *p(weight=130|male), p(fs=8|male), p(height=6|female), p(weight=130|female), p(fs=8|female).***

Normal Distribution Naïve Bayes Continuous Variables

- **We get the following results:**

$$
\begin{aligned}
&p(\text{weight} \mid \text{male}) = 5.9881 \cdot 10^{-6}\\
&p(\text{foot size} \mid \text{male}) = 1.3112 \cdot 10^{-3}\\
&\text{posterior numerator (male)} = \text{their product} = 6.1984 \cdot 10^{-9}\\
&P(\text{female}) = 0.5\\
&p(\text{height} \mid \text{female}) = 2.2346 \cdot 10^{-1}\\
&p(\text{weight} \mid \text{female}) = 1.6789 \cdot 10^{-2}\\
&p(\text{foot size} \mid \text{female}) = 2.8669 \cdot 10^{-1}\\
&\text{posterior numerator (female)} = \text{their product} = 5.3778 \cdot 10^{-4}
\end{aligned}
$$

- **Since *p(female|…) > p(male|…)*, we classify this person as "female".**

Multinomial Naïve Bayes Frequencies

- **In multinomial Naive Bayes, we consider an instance x to consist of a vector of frequencies $(x_1, x_2, \ldots, x_n)$ where each x_i is the frequency of some event i occurring.**
  - E.g. in a document classification problem, x_i is the number of times word i appears.
  - Bag of words assumption:
    - ✓ **A document is simply a collection of words.**
    - ✓ **We ignore sentence structure, context, etc.**
  - The vector x is therefore a histogram of word frequencies.

Multinomial Naïve Bayes Frequencies

- **We compute the probability of event *i* occurring in class *c_k* by taking:**

$$p_{ik} = \frac{x_i}{\sum_{j=1}^{n} x_k}$$

Where x_i is the frequency for *i* in class c_k in the training data. Thus if event *i* is the word "hello", then x_i is the number of times "hello" appears in class c_k, while the denominator is the total number of word occurrences in the class.

- **The probability $p(x|c_k)$ of a sample histogram *x* occurring in class c_k is given by:**

$$p(x|c_k) = \frac{(\sum_{i=1}^{n} x_i)!}{\prod_{i=1}^{n} x_i!} \prod_{i=1}^{n} p_{ik}^{x_i}$$

- **To find which class *k* our instance x belongs to we find:**

$$argmax_k \; p(c_k)p(x|c_k)$$

Multinomial Naïve Bayes Frequencies

- **Issue 1: Raw frequencies in document classification face some problems:**
  - Bias towards longer documents.
  - Bias towards "connectors" like "the" because they occur much more frequently.
  - We can fix the first problem by using term-frequency inverse document frequency (tf.idf) instead if raw word counts:

$$x_i = \log(tf_{ik} + 1)\log(\frac{D}{d_{tf}})$$

  - Here tf_{ik} is the raw "term frequency" of i occurring in class k, while D is the total number of documents, and d_{tf} is the total number of documents containing i.
- **Issue 2: Zero frequencies**
  - If $x_i=0$ in class c_k, then $p_{ik} = 0$ and $p(x|c_k)$ becomes 0.
  - Laplace smoothing: Add 1 to every x_i so that no p_{ik} is 0.

Bernoulli Naïve Bayes
Boolean Features

- **Sometimes, rather than maintaining a count of the number of times event *i* occurs in class c_k, we are only interested in whether it occurs at all.**
  - In such cases we use a Bernoulli distribution instead of multinomial.
  - Our instance *x* is a vector $(x_1, x_2, \ldots, x_n)$ where $x_i=1$ if event *i* occurs in class c_k, and $x_i=0$ otherwise.
- **We begin by calculating p_{ik} the probability that *i* occurs in class c_k**

$$p_{ik} = \frac{n_{ik}}{N_k}$$

Where n_{ik} is the number of training examples where *i* occurs in class c_k, and N_k is the total number of training examples in class c_k.

- **The probability of a Boolean vector *x* occurring in class c_k is then:**

$$p(x|c_k) = \prod_{i=1}^{n} p_{ik}^{x_i}(1 - p_{ik})^{(1-x_i)}$$

Naïve Bayes Hands-on

Start up Jupyter and load the "Naïve Bayes.ipynb" notebook.

SWS3009 Embedded Systems and Deep Learning

CLASSIFICATION: DECISION TREES

Decision Trees

- **A decision tree is a tree-like structure for making decisions:**

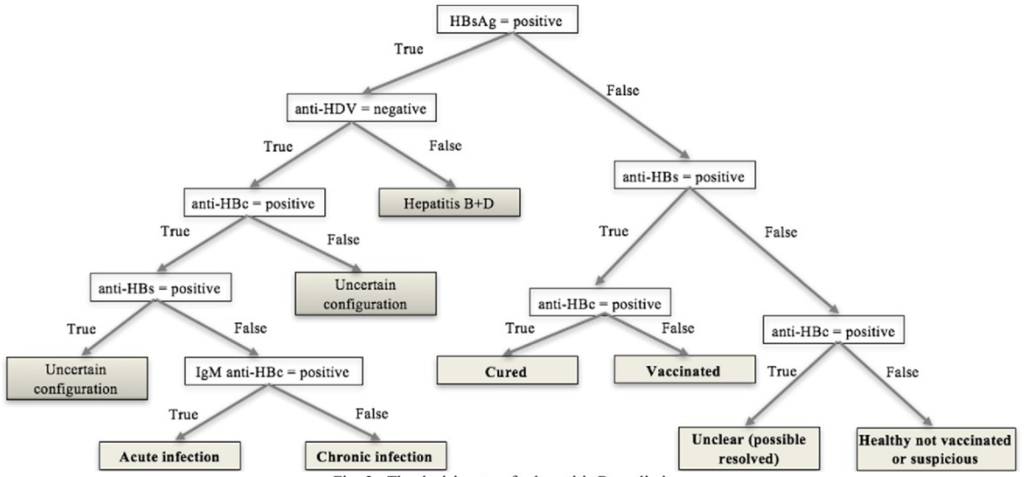
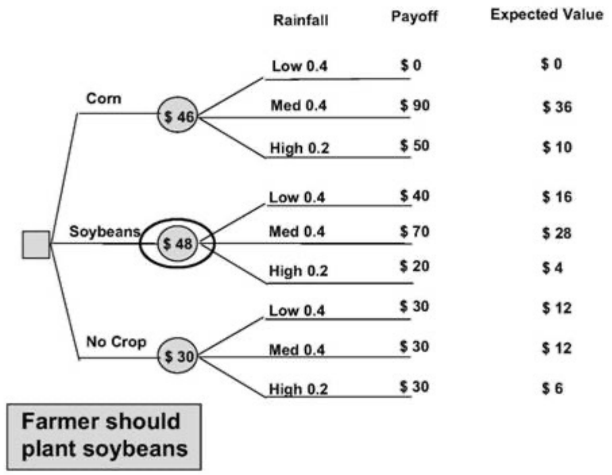


Fig. 2. The decision tree for hepatitis B predictions

- **A decision tree is appealing because:**
  - It provides a straightforward way to make decisions.
  - It provides a way to explain decisions that have already been made.

Decision Trees

- **Decision trees sometimes have probability values attached to them, allowing you to compute expected values of outcomes:**



- **If there's a 40% probability of low rainfall, a 40% probability of medium rainfall and 20% probability of high rainfall, with the payoff per acre of farmland as indicated in the "payoff" column, this chart helps the farmer to decide which is the best crop to plant (soybeans), based on expected revenue.**

Decision Trees

- **Question:**
  - How do you derive these trees?
- **Several ways:**
  - Guesswork based on 'expert' opinion.
  - Based on historical data.
- **We will look at the second way.**
- **Big shoutout:**

Most of the materials from this section are taken, with permission, from Sefik Ilkin Serengil's excellent blog on machine learning. If you are interested face recognition and other cool stuff, be sure to go over to:

https://sefiks.com/category/machine-learning/

Actual decision tree materials are at: https://sefiks.com/2017/11/20/a-step-by-step-id3-decision-tree-example/

Deriving Decision Trees from Data

- **Entropy: The amount of uncertainty in data.**
  - Let's suppose we have a system S with two classes of data A and B, with n_A and n_B items respectively, and $n = n_A + n_B$.
  - By classic probability, $p(A) = \frac{n_A}{n}$ and $p(B) = \frac{n_B}{n}$. The entropy H(S) of system S is given by:

$$H(S) = -p(A)\log_2 p(A) + (-p(B)\log_2 p(B))$$

  - In the "clearest" case, $n_A = n$ and $n_B = 0$. I.e. all items belong to class A (note: this applies to the converse as well). $\log_2 0$ is infinite and we do not consider it (See L'Hopital Rule for a more rigorous solution). Then:

$$H(S) = -1 \times \log_2 1$$
$$= 0$$

  - In the "worst" case, $n_A = n_B = n/2$. Then $p(A) = p(B) = 0.5$, and:

$$H(S) = -0.5 \times \log_2 0.5 + (-0.5 \times \log_2 0.5)$$
$$= 1$$

Deriving Decision Trees from Data

- **We see that $0 \le H(S) \le (1)$, with H(S) = 0 in the "purest" case when all items belong to one category, and H(S) = 1 in the "worst" case that all items are evenly distributed over all categories.**
- **In general, for *N* categories:**

$$H(S) = \sum_{i=1}^{N} -p(i)log_2 p(i)$$

- **Here $0 \le H(S) \le log_2 N$, with *H(S) = 0* when all items belong to one category, and *H(S) = log_2 N* when all items are evenly distributed across all *N* categories.**
- **For simplicity we will look at the older ID3 algorithm. We will discuss the newer C4.5 algorithm very briefly at the end of this section.**

The ID3 Algorithm

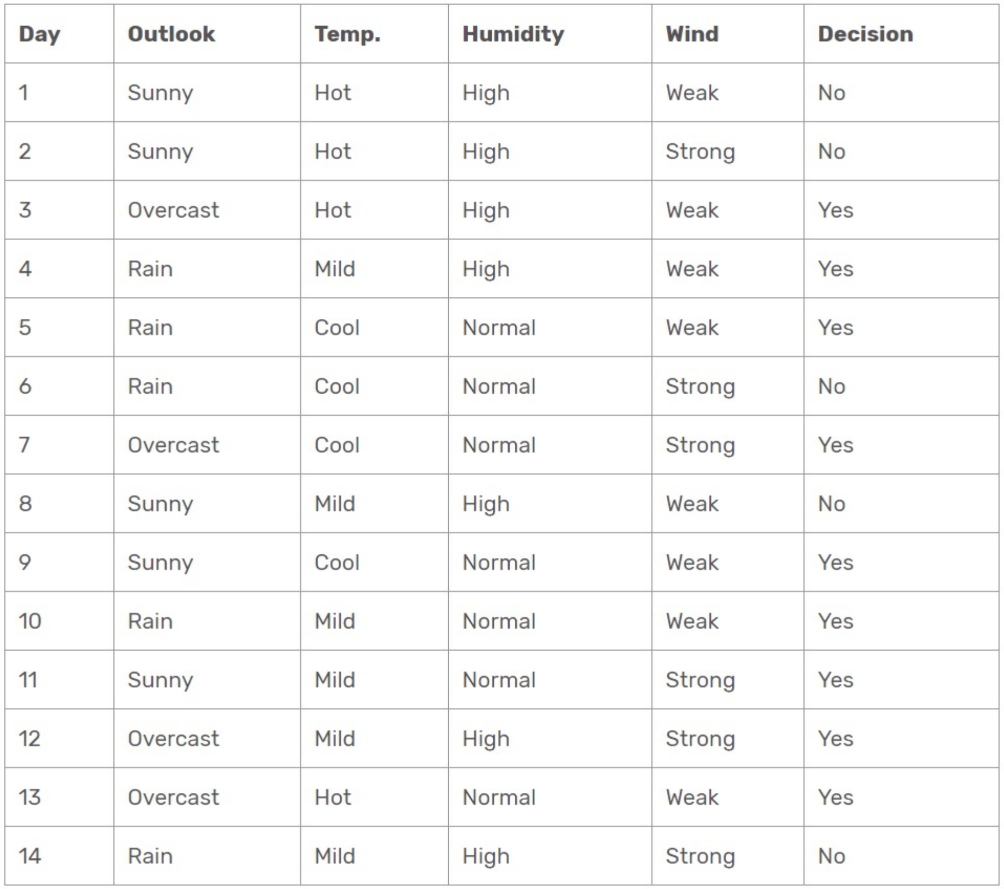
- **The basis of this algorithm is that we choose decision criteria that reduce uncertainty as much as possible. This reduction is called the "gain" and is defined as:**

$$\boldsymbol{gain(S|A) = H(S) - \sum_{x \in v(A)} p(A = x) \sum_{y \in d} H(S|A = x)}$$

- **In this equation, *gain(S|A)* is the gain in information (or reduction in uncertainty) if we chose to decide over criteria *A*, v(A) is the set of values that category A can take, and *H(S|A=x)* is the entropy of the system when we decide based on A taking the value of x, computed using probabilities of each decision in the set of decisions d.**
  - Sounds confusing, so let's look at an example.
    - ✓**You want to play tennis outdoors. You look at the weather outlook, the temperature, humidity and wind to decide.**
    - ✓**So hard to decide! So you look at your friend Bob's pattern of decision making, and try to learn from him.**

The ID3 Algorithm

- **Bob's playing history for the past 14 days, based on weather, is shown in this table:**



Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

The ID3 Algorithm

- **Let's begin by computing *H(S)*. There are two categories (Yes and No), and 9 instances of Yes and 5 of No. The entropy then is:**

$$\boldsymbol{H(S) = -\frac{9}{14}.log_2\left(\frac{9}{14}\right) + \left[-\frac{5}{14}.log_2\left(\frac{5}{14}\right)\right]}$$

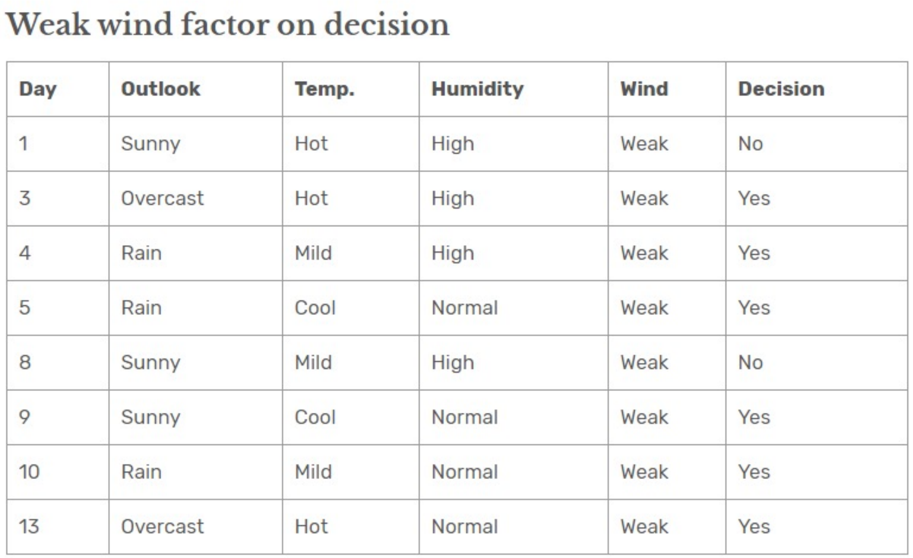
$$\mathbf{= 0.940}$$

- **Ok here comes the really painful bit: We need to find the decision factor that results in the *largest* gain. Yes, we need to go through each of them in turn. Let's begin with wind:**
  - There are two wind values: Weak and Strong.
  - We compute the entropies for each decision *y* from the set of possible decisions d, for each possible value of wind. In our case d = {Yes, No}.

$$H(S\,|Wind = Weak) = \sum_{y \in d} p(y|Wind = Weak)\log_2 p(y|Wind = Weak)$$

$$H(S\,|Wind = Strong) = \sum_{y \in d} p(y|Wind = Strong)\log_2 p(y\,|Wind = Strong)$$

The ID3 Algorithm



Weak wind factor on decision

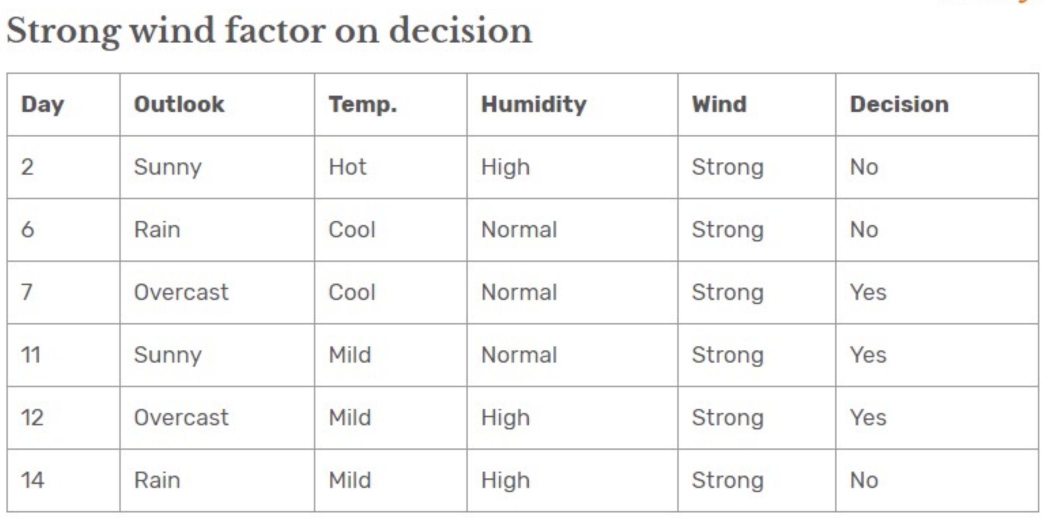
Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
13	Overcast	Hot	Normal	Weak	Yes

- **Let’s look at the decisions made during weak wind:**
  - There are 6 “Yes” decisions and 2 “No”. From this we can compute H(S|A=weak):

$$H(S|A = Weak) = -p\left(\frac{2}{8}\right).\log\left(\frac{2}{8}\right) - p\left(\frac{6}{8}\right)\log\left(\frac{6}{8}\right)$$

$$= 0.811$$

The ID3 Algorithm



Strong wind factor on decision

Day	Outlook	Temp.	Humidity	Wind	Decision
2	Sunny	Hot	High	Strong	No
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
14	Rain	Mild	High	Strong	No

- **Let's look at decisions made during strong wind:**
  - There are 3 "No" decisions and 3 "Yes" decisions. We can compute H(S|A=Strong)

$$H(S|A = Strong) = \left(\frac{3}{6}\right)\log_2\left(\frac{3}{6}\right) - \left(\frac{3}{6}\right)\log_2\left(\frac{3}{6}\right)$$

The ID3 Algorithm

- **Now we can compute our gain. Since there are 8 "weak wind" and 6 "strong wind" records:**
  - $p(Wind = Weak) = \frac{8}{14}, p(Wind = Strong) = \frac{6}{14}$
- Thus the Gain is now:

$$\begin{aligned}&Gain(S|Wind)\\&= H(S) - (P(Wind = Weak).H(S|Wind = Weak)\\&+ P(Wind = Strong).H(S|Wind = Strong)\end{aligned}$$

$$= 0.940 - \left(\frac{8}{14}.0.811 + \frac{6}{14}.1\right)$$

$$= 0.048$$

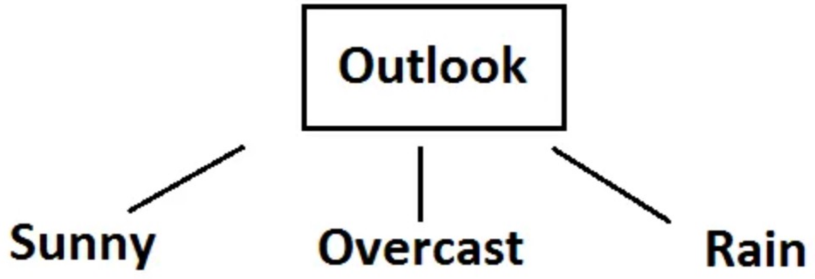
The ID3 Algorithm

- **We repeat for all the columns and get:**

Factor	Gain
Wind	0.048
Outlook	0.246
Temperature	0.029
Humidity	0.151

- **The factor with the highest gain is 0.246, so we pick that as the root and get the following tree:**

The ID3 Algorithm



- **There are three outcomes for Outlook: Sunny, Overcast, Rain. Let's look at the decision history for these three outcomes.**
  - First we look at Overcast. We find that the decision is always yes. We can stop here.

Day	Outlook	Temp.	Humidity	Wind	Decision
3	Overcast	Hot	High	Weak	Yes
7	Overcast	Cool	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes

The ID3 Algorithm

- We look at the outcome for Sunny:

Sunny outlook on decision

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

- We work out the gains for temperature, humidity and wind:

Factor	Gain
Temperature	0.570
Humidity	0.970
Wind	0.019

The ID3 Algorithm

- **Here humidity has the highest gain. There are two possible outcomes: "High" and "Normal". We can look at the decisions made for both when it is sunny:**

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No

Day	Outlook	Temp.	Humidity	Wind	Decision
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

- **The decision is straightforward: Yes if humidity is normal, no if high.**

The ID3 Algorithm

- **Now let's look at decisions when it is raining:**

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

- **Let's look at the gains for the remaining factors:**

Factor	Gain
Temperature	0.019973
Humidity	0.019773
Wind	0.9710

The ID3 Algorithm

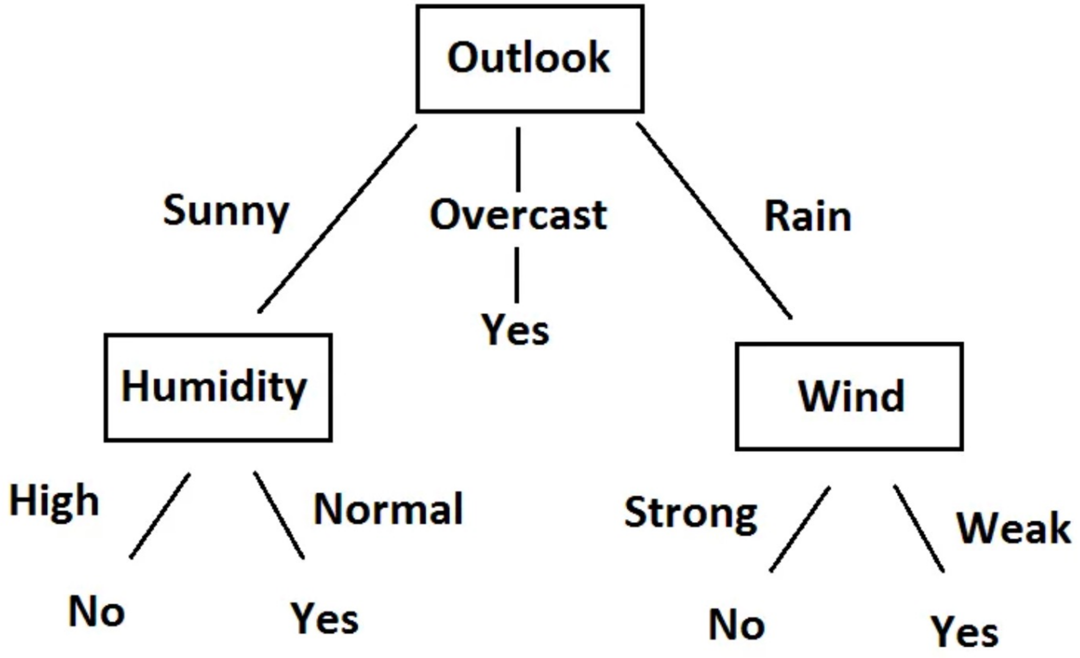
- **Here Wind has the highest gain. There are two values: Strong and Weak. We can look at the decisions for both:**

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes

Day	Outlook	Temp.	Humidity	Wind	Decision
6	Rain	Cool	Normal	Strong	No
14	Rain	Mild	High	Strong	No

The ID3 Algorithm

- **So when it rains, we will always play when the wind is weak, and never when it is strong. We are done! This is how we decide whether or not to play tennis (or at least this is how Bob decides):**



The C4.5 Algorithm, Regression Trees

- **The ID3 Algorithm has one drawback:**
  - It only works with "nominal" values – i.e. each factor has values that are clear categories. It doesn't work with continuous values.
- **The C4.5 algorithm addresses this. In addition the gains of each factor are also scaled by their entropies.**
  - You can find the C4.5 algorithm described at: https://sefiks.com/2018/05/13/a-step-by-step-c4-5-decision-tree-example/
- **Here we use decision trees for classification. They can also be used for regression, but this will be outside the scope of this course. Please see https://sefiks.com/2018/08/28/a-step-by-step-regression-decision-tree-example/**

Decision Trees Hands-on

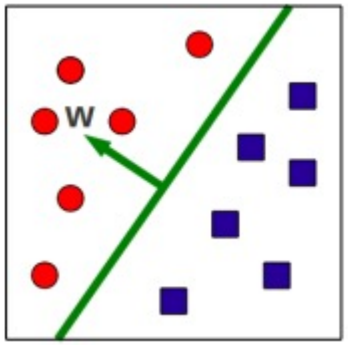
Start up Jupyter and load the “DecisionTrees.ipynb” notebook.

SWS3009 Embedded Systems and Deep Learning

CLASSIFICATION: SUPPORT VECTOR MACHINES

Support Vector Machines
Separation Hyperplanes

- **A hyperplane is a *d* dimension plane defined by its normal vector *w*.**
  - In 2D a hyperplane is a straight line, in 3D it is a standard plane.
  - The normal ***w*** is orthogonal to all points on the plane. I.e. for any point ***p*** on the plane, $\boldsymbol{w^T p}$ is 0.
  - The hyperplane separates the *d*-dimension space into two halves.
  - Hyperplanes always pass through the origin:
    - ✓**This restricts its ability to partition the space.**
    - ✓**To fix this we add a *d*-dimension bias *b*.**



- **Our goal:**
  - To learn ***w*** and ***b*** such that given an input ***x***:

$$w^T x + b > 0 \; if \; x \; belongs \; to \; class \; A$$
$$w^T x + b < 0 \; if \; x \; belongs \; to \; class \; B$$

Support Vector Machines Assumptions

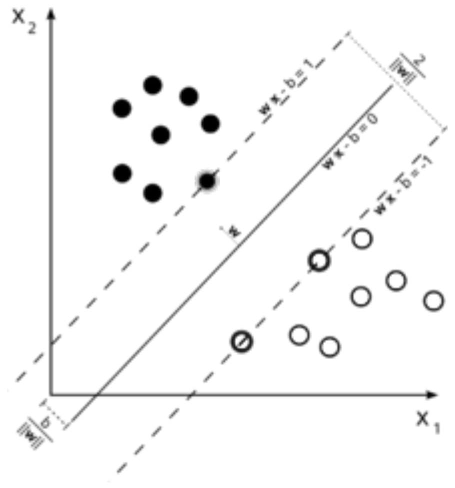
- **A “Support Vector Machine” or SVM is a classifier that uses two hyperplanes to separate points between two classes:**

$$w_1^T x + b_1 = 1$$

- **Anything above this hyperplane will be classified as a “1” (class A, class 1, etc)**

$$w_2^T x + b_2 = -1$$

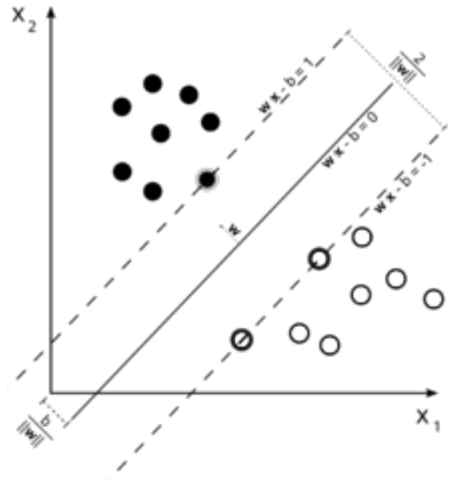
- **Anything below this line will be classified as a “-1” (class B, class 2, etc.)**



Support Vector Machines Assumptions

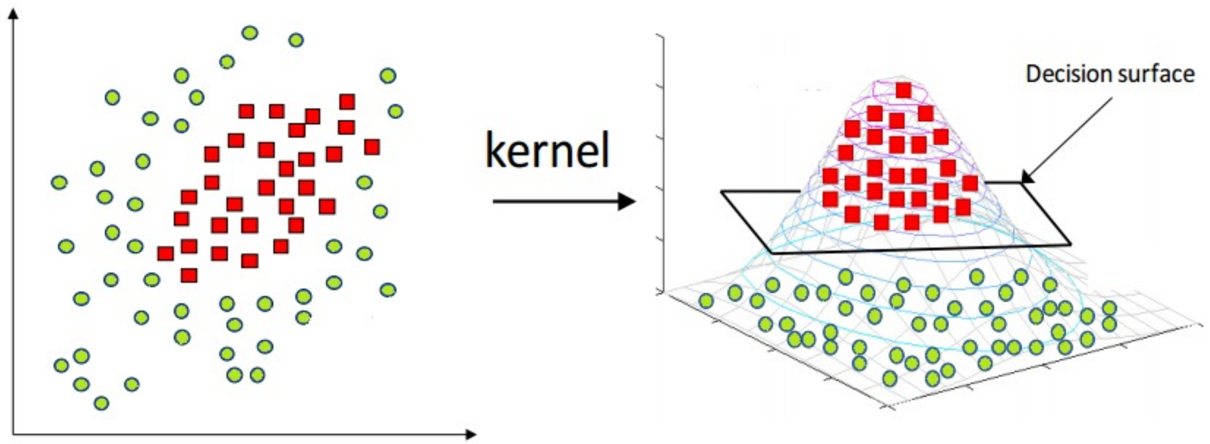
- **These two hyperplanes must be as far apart as possible:**
  - This means that they will be determined by the points in two different classes (1 and -1) that are closest together.
  - These points are called “support vectors”.
- **In practice we want a hyperplane $w^T + b$ that is in the middle of these two hyperplanes. This maximizes the margin $\gamma = \frac{1}{||w||}$ between the classes.**
- **This is equivalent to maximizing ||w||, subject to:**

$$y(x_i . w) \geq 1$$



Support Vector Machines
Non-Linearly Separable Points

- **The standard SVM can only classify linearly separable points. For points that are not linearly separable, SVMs can apply a "kernel trick":**
  - ▪Choose a kernel function that maps the points to a higher dimension.
  - ▪If properly chosen the higher dimension points should become linearly separable.



Support Vector Machines Parameter Estimation

- **Parameter estimation begins with choosing a "loss function". The simplest is "empirical risk":**

$$\boldsymbol{R_{emp} = \frac{1}{m} l(f(x_i, (w, b)), y_i)}$$

▪Here *m* is the number of samples, *f(.)* is the model we are trying to estimate, *w* are the weights of our hyperplane, b is the bias, and x_i and y_i are the current input and label (class) respectively. *l()* is our "loss function" defined as:

$$l(y_{est}, y_i) = 1\ if\ y_{est} \neq y_i, 0\ otherwise$$

- **We minimize R_{emp} subject to $min_i|w.x_i|=1$.**
- **This is a difficult problem to solve, and is done using Lagrangians (Outside the scope of this course).**

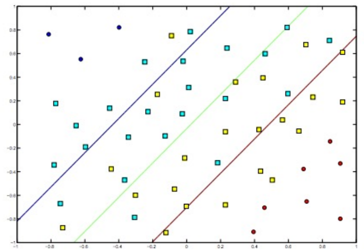
Support Vector Machines Configuration

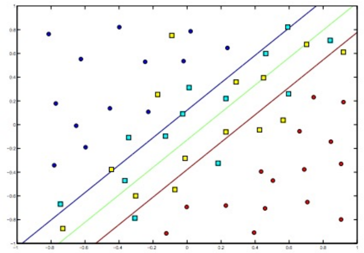
- **There are several things we need to choose when using Linear SVMs:**
  - Loss function:
    - ✓**Hinge Loss: Loss function used for linearly separable classes.**
    - ✓**Logistic: Produces probability estimates of the classifications.**
    - ✓**Perceptron: Loss function used by Perceptrons**
    - ✓**Modified Huber: More tolerance to outliers (see later).**

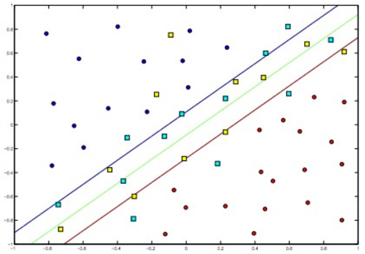
  - Training Rate (alpha): Controls how fast the SVM learns. Higher values can lead to poorer results, lower values can lead to slow training.

Support Vector Machines Configuration

- **Regularization: This controls how we penalize mis-classified information (i.e. information that should be in one class, but occurs in the space of another class - outliers):**
  - Higher penalty: Possibly more overfitting – SVM can only understand and correctly classify training data and nothing else.
  - Lower penalty: Underfitting – SVM performs poorly even on training data.
  - The idea is to choose a regularization that allows *some* misclassified information, but not too much.
    - ✓**L1 – Adds a penalty term that uses the absolute value of the model parameters. Results in a model with fewer parameters.**
    - ✓**L2 – Adds a penalty term that uses the square of the value of the model parameters. Produces denser models.**
    - ✓**Elastic: Combines L1 and L2.**


($\lambda = 0.1$)


($\lambda = 1$)


($\lambda = 1000$)

SVM Hands-on

Start up Jupyter and load the "SVM.ipynb" notebook.

Summary

- **Statistical methods make use of statistical assumptions about the data to build models:**
  - Linear Regression: There is strong correlation between the variables.
  - Naïve Bayes: Statistical independence between variables.
  - Decision Trees: Clear decision paths inferred from past decision data.
  - Support Vector Machines: There exists a hyper-plane that can separate points.
- **Statistical methods are useful:**
  - Simple to build, fast to train.
  - Can be more useful than neural networks when the data is relatively simple.
  - Can be good to "test" data to see if it is possible to even build machine learning models for it.
    - ✓ **E.g. in the temperature-location example we did in Lab 1, there is little deviation in temperature across the whole of Singapore.**
    - ✓ **Unlikely to be able to train a machine learning model to predict temperature based on location – indeed we failed in doing so.**