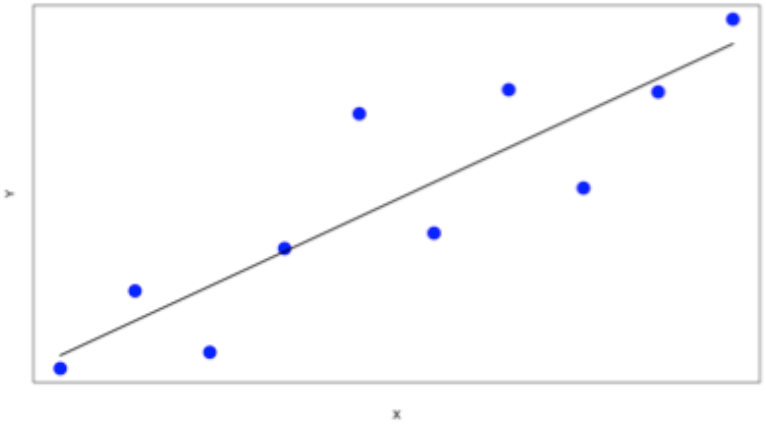

20T2
Comp9417
Review 1

Comp9417
Linear Regression

In linear regression, we assume there is a linear relationship between the output and feature(s)/input variable(s); this means the expected value of the output given an input, $E[y|x]$ is linear in input x



Example with one variable: $\hat{y} = bx + c$

Problem: given the data, estimate b and c

Comp9417

Closed-form Solution

Univariate linear regression: (only one independent variable)

Goal: fit a line such that $\hat{y} = \theta_0 + \theta_1 x$

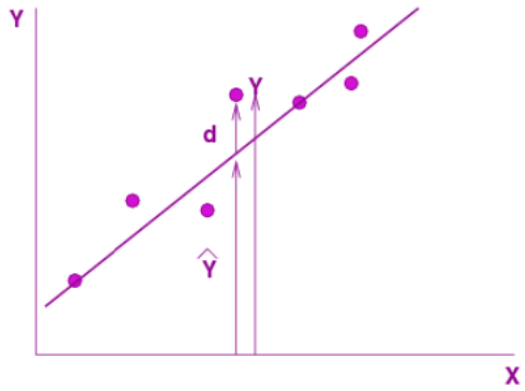
How: minimize $J(\theta) = \sum_j (y_j - \hat{y}_j)^2$ (Least squares estimator)

- If we expand the explicit solution we saw before, we can see:

$$\theta_1 = \frac{cov(x, y)}{var(x)}$$

Where $cov(x, y)$ is the covariance of x and y

- and $\theta_0 = \bar{y} - \theta_1 \bar{x}$



Comp9417
Covariance

Covariance is a measure of relationship between two random variables:

$$cov(x,y) = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{N-1} = \frac{(\sum_i x_i y_i) - N\bar{x}\bar{y}}{N-1}$$

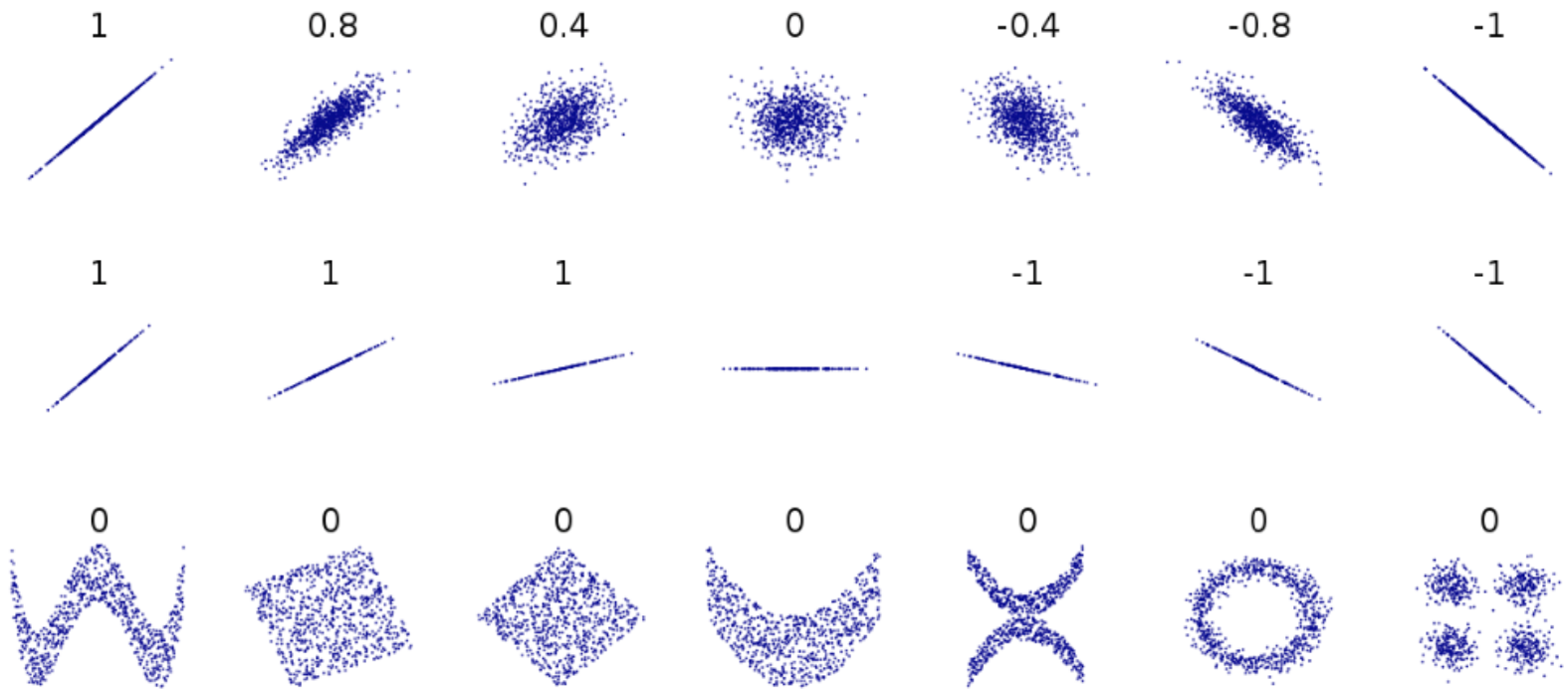
Correlation is a measure to show how strongly a pair of random variables are related

- The formula for computing correlation between x and y is:

$$r = \frac{cov(x,y)}{\sqrt{var(x)}\sqrt{var(y)}}$$

This is also called *Pearson's correlation coefficient*

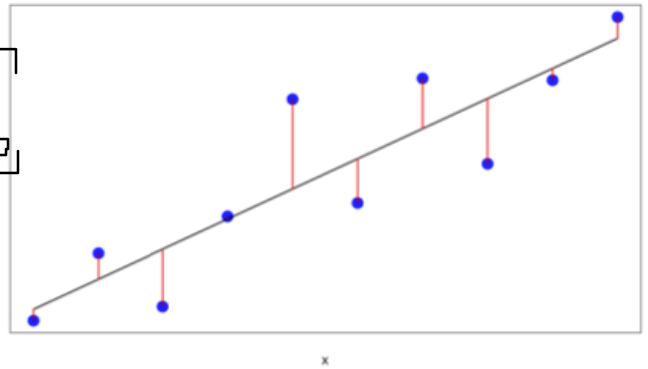
Comp9417
Correlation



Comp9417
Closed-form solution

Another important point to note is that the **sum of the residuals** of the least square **is zero**

$$\sum_{j=1}^{m}(y_j - x_j^T\theta) = 0$$



While this property is intuitively appealing it is worth keeping in mind that it also makes linear regression susceptible to *outliers:* points that are far removed from the regression line, often because of the measurement error.

Comp9417
Gradient Descent

$$\frac{\partial J(\theta)}{\partial \theta_i} = 2(y_j - h_\theta(x_j)) \frac{\partial (y_j - h_\theta(x_j))}{\partial \theta_i}$$
$$= 2(y_j - h_\theta(x_j))(-x_{ji})$$

If we focus on only one sample out of m samples, then the cost function is:

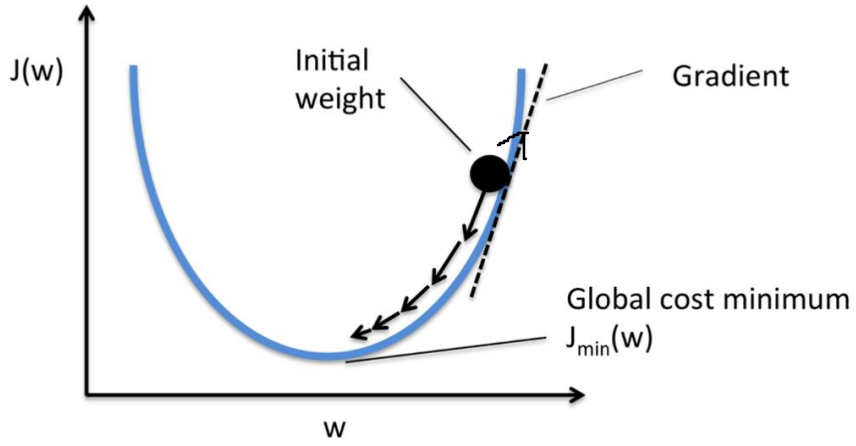
$$J(\theta) = (y_j - h_\theta(x_j))^2 = (y_j - \sum_{i=1}^{n} x_{ji}\theta_i)^2$$

Taking the derivative will give:

$$\frac{\partial}{\partial \theta_i} J(\theta) = 2(h_\theta(x_j) - y_j)x_{ji}$$

So, for a **single** training sample, the update rule is:

$$\theta_i := \theta_i + \alpha(y_j - h_\theta(x_j))x_{ji} \quad \text{(for every } i\text{)}$$



The update rule for squared distance is called "**Least Mean Squares**" (LMS), also known as **Widrow-Hoff**

Comp9417

Gradient Descent

1. Batch Gradient Descent:

$$\theta_i = \theta_i + \alpha \sum_{j=1}^{m} (y_j - h_\theta(x_j)) x_{ji} \quad \text{(for every } i\text{)}$$

Replace the gradient with the sum of gradient for all samples and continue until convergence.

2. Stochastic Gradient Descent:

$$for\ j = 1\ \ to\ \ m\ \{$$
$$\theta_i = \theta_i + \alpha (y_j - h_\theta(x_j)) x_{ji} \quad \text{(for every } i\text{)}$$
$$\}$$

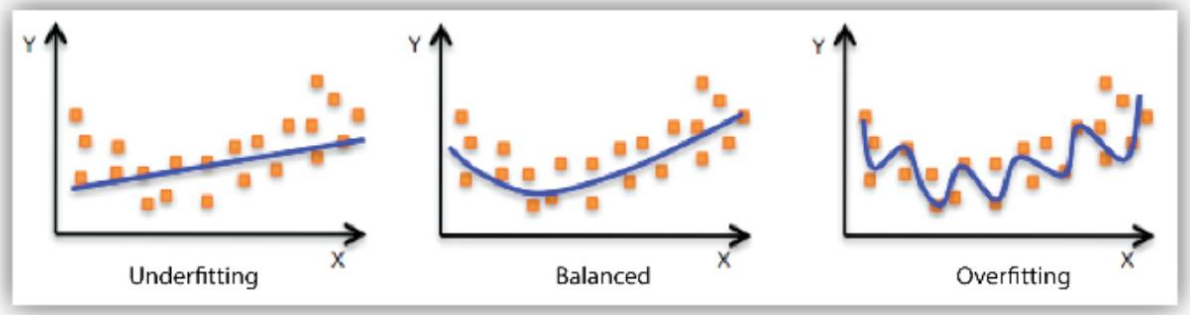
Repeat this algorithm until convergence.

Comp9417

Linear Regression for curve shapes

- In this example we can predict the output with different models:
- $\hat{y} = \theta_0 + \theta_1 x_1$
- $\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2,\ x_2 = x_1^2. \rightarrow\ \hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2$
- $\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^3, x_2 = x_1^2,\ x_3 = x_1^3 \rightarrow\ \hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$

Comp9417
Regularization



- MSE as a cost function, given data $(x_1, y_1), \ldots, (x_m, y_m)$

$$J(\theta) = \sum_j (y_j - h_\theta(x_j))^2 + \lambda \sum_i \theta_i^2$$

- Parameter estimation by optimisation will attempt to values for $\theta_0, \ldots, \theta_n$ s.t. $J(\theta)$ is a minimum

- Similar to before, this can be solved by gradient descent or take the derivatives and set them to zero

Comp9417
Regularization

The multiple least-squares regression problem is an optimisation problem, as we saw, and can be written as:

$$\theta^* = \arg\min_{\theta}(y - X\theta)^T(y - X\theta)$$

The regularised version of this is then as follows:

$$\theta^* = \arg\min_{\theta}((y - X\theta)^T(y - X\theta) + \lambda||\theta||^2$$ ***ridge regression***.

$$\lambda\sum_i|\theta_i|$$ ***lasso regression***.

Where $||\theta||^2 = \sum_i \theta_i^2$ is the squared norm of the vector θ, or equivalently, the dot product $\theta^T\theta$; λ is a scalar determining the amount of regularisation.

Comp9417
Sample

Exercise: To make sure you know the process, try to solve the following loss function for linear regression with a version of "$L2$" regularization:

$$\mathcal{L} = \frac{1}{N}\sum_{n}(y_n - (w_0 + w_1 x_n))^2 + \frac{\lambda(w_1)^2}{2}$$

Answer

$$\begin{aligned} \widehat{w_0} &= E(y) - w_1 E(x) \\ \widehat{w_1} &= \frac{E(xy) - E(x)E(y)}{E(x^2) - (E(x))^2 + \lambda/2} \end{aligned}$$

Comp9417
Sample

x	y
4	2
6	4
12	10
25	23
29	28
46	44
59	60

We wish to fit a linear regression model to this data, i.e. a model of the form:

$$\hat{y}_i = w_0 + w_1 x_i.$$

We consider the Least-Squares loss function:

$$L(w_0, w_1) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2,$$

where n is the total number of training points.

Comp9417
Sample

(a) Derive the least squares estiamtes of w_0 and w_1 (showing their working), and compute them for the provided data.

x	y
4	2
6	4
12	10
25	23
29	28
46	44
59	60

$$w_0 = \overline{y} - w_1\overline{x}, \quad w_1 = \frac{\overline{xy} - \overline{x}\ \overline{y}}{\overline{x^2} - \overline{x}^2}$$

$$w_0 = -2.4570, \quad w_1 = 1.0398.$$

Comp9417
Sample

(b) Based on your linear model, what is the prediction for a test point $x_\star = 50$?

$50 w_1 + w_0$

(c) Consider a new training point, $(x_{\text{new}}, y_{\text{new}}) = (45, 0)$. What are the new least squares parameters if we include all training data (including this new point)?

计算新的w0 w1

(d) The new training point in (c) can be considered to be what kind of point? What does such a point mean for your estimated parameters? How could you remedy the situation?

noise 直接数据可视化，发现 这个点与正常分布很远

或者使用MAE lost function 采取绝对值的求和，减少noise point 对数据的影响

Comp9417

Sample

(f) Compute the derivatives of the following functions

1. $f(x) = x^2 \ln(x)$
2. $g(x) = (1 + 2x^4)^3$
3. $h(x) = \frac{2\ln(x)+4x}{x^3}$

$$(u \pm v)' = u' \pm v' \qquad (uv)' = u'v + uv' \qquad \left(\frac{u}{v}\right)' = \frac{u'v - uv'}{v^2}$$

1. $f'(x) = 2x \ln(x) + x$
2. $g'(x) = 24x^3(1 + 2x^4)^2$
3. $h'(x) = \frac{\left(\frac{2}{x}+4\right)x^3-(2\ln(x)+4x)3x^2}{x^6} = \frac{2-6\ln(x)-8x}{x^4}$.

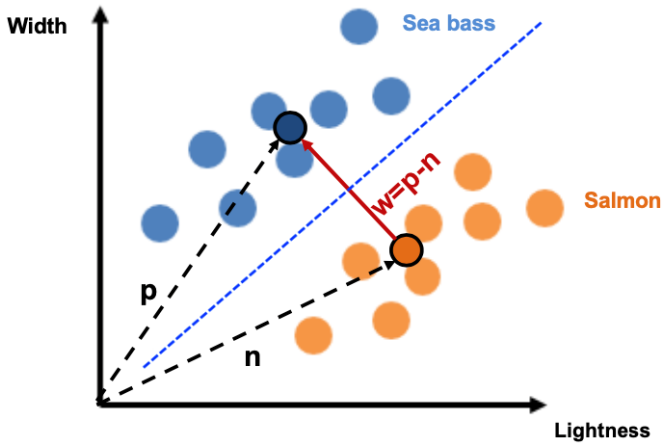
Comp9417

Linear classification

- We find the line that separates the two class: $ax_1 + bx_2 + c = 0$
- We define a weight vector $w^T = [a, b]$, $x^T = [x_1, x_2]$
- So the line can be defined by $x^T w = -c = t$
- w is perpendicular to decision boundary (in direction of positive class)
- t is the decision threshold (if $x^T w > \mathrm{t}$ then x belongs to positive class and if $x^T w < \mathrm{t}$ then x belongs to negative class)

Comp9417
Linear classification

The basic linear classifier constructs a decision boundary by half-way intersecting the line between the positive and negative centres of mass.



The basic linear classifier is described by the equation $x^T w = t$, and $w = p - n$

As we know, $\frac{p+n}{2}$ is on the decision boundary, so we have:

$$t = (\frac{p+n}{2})^T . (p-n) = \frac{||p||^2 - ||n||^2}{2}$$

Where $||x||$, denotes the length of vector x

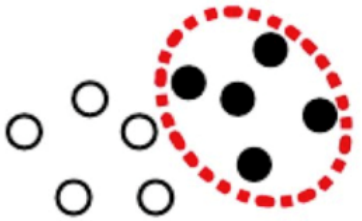
$x^T \cdot (p - n) =$

Comp9417
Classification

可以产生点并且产生分布

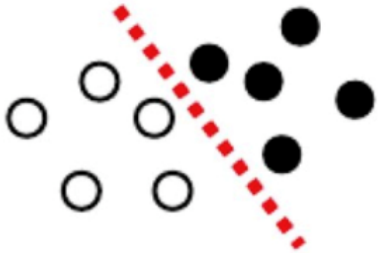
Generative algorithm: builds some models for each of the classes and then makes classification predictions based on looking at the test example and see it is more similar to which of the models.

- Learns $p(x|y)$ (and also $p(y)$, called class prior)
- So we can get $p(x, y) = p(x|y)p(y)$
- It learns the mechanism by which the data has been generated



Discriminative algorithm: Do not build models for different classes, but rather focuses on finding a decision boundary that separates classes

- Learns $p(y|x)$



只能判断但是不能产生分布

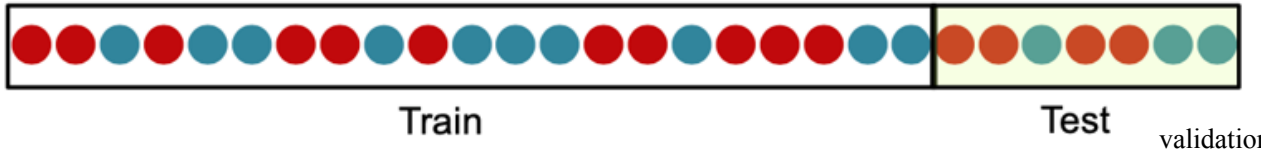
Comp9417
Train Valid & Test

- **Train Data:** the data that we use to learn our model and its parameters
- **Validation Data**: The data (unseen by the model) used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters.
- **Test Data:** unseen data by the model that we use to test the model and shows how well our model generalizes. In practice, we don't know the ground truth (label) for this data

Comp9417

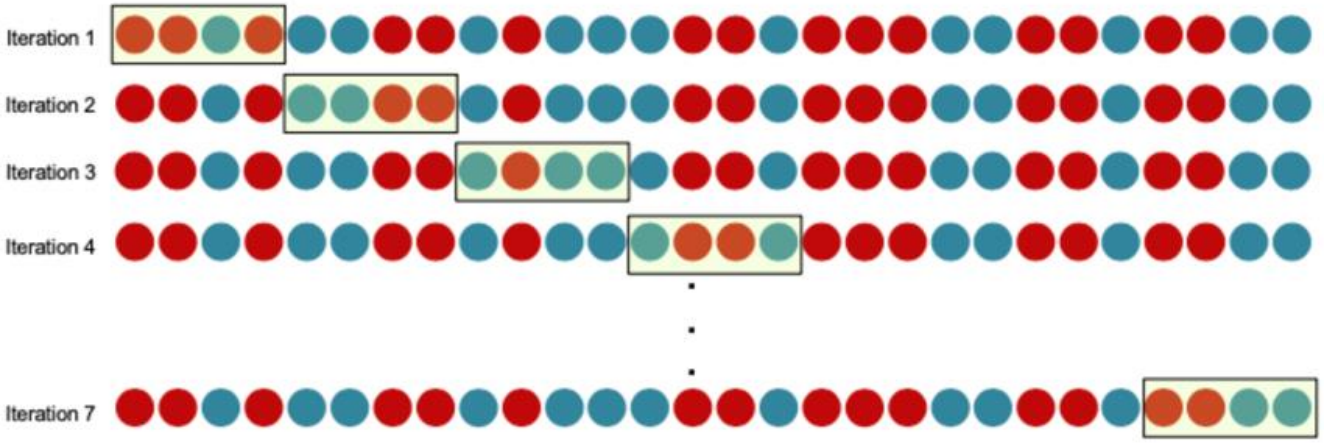
Cross-validation

There are certain parameters that need to be estimated during learning. We use the data, but NOT the training set, OR the test set. Instead, we use a separate *validation* or *development* set.



validation

K-fold Cross Validation



Comp9417
Data Types

In Machine Learning world, in general two types of data is defined:

- **Numerical**: Anything represented by numbers (e.g., integer , floating point)
- **Categorical**: everything that is not numerical (e.g. discrete labeled groups)

In general, for machine learning algorithms, data has to be represented in numeric form

Comp9417

Normalization

- Different attributes measured on different scales (for example one attribute/feature may have a range of $[0,100]$ and another have a range of $[-1,1]$)
- Need to be *normalized* (why ?)

$$x'_{jr} = \frac{x_{jr} - \min(x_{jr})}{max(x_{jr}) - \min(x_{jr})}$$

where x_{jr} is the actual value of attribute/feature r and x'_{jr} is the normalised value.

- Nominal attributes: distance either 0 or 1

Comp9417

Evaluation of classification

- For two-class prediction case:

Actual Class	Predicted Class	
	Positive	Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

- $acc = \frac{1}{|Test|} \sum_{x \in Test} I[\hat{c}(X) = c(X)]$ $\frac{TP+TN}{TP+FN+FP+TN}$
- $Precision = \frac{TP}{TP+FP}$
- $Recall = \frac{TP}{TP+FN}$
- $F_1 = 2.\frac{precision.recall}{precision+recall}$
- $AUC - ROC\ curve$

Comp9417

Missing Value

Deleting samples with missing values:

- Pros:
  - A robust and probably more accurate model
- Cons:
  - Loss of information and data

Replacing the missing value with mean/median/mode:

- Pros:
  - When the data size is small, it is better than deleting
- Cons:
  - Imputing the approximations add variance and bias

Assigning a unique category:

- Pros:
  - No loss of data
- Cons:
  - Adds less variance

Comp9417
Missing Value

Predicting the missing values:

- Pros:
  - Yields unbiased estimates of the model parameters
- Cons:
  - Bias also arises when an incomplete conditioning set is used for a categorical variable

Using algorithms that support missing values:

- Pros:
  - Correlation of the data is neglected
- Cons:
  - Is a very time-consuming process and it can be critical in data mining where large databases are being extracted

Comp9417
Distance

Minkowski distance If $\mathcal{X} \to \mathbb{R}^d, x, y \in \chi$, the Minkowski distance of order $p > 0$ is defined as:

$$Dis_p(x,y) = (\sum_{j=1}^{d} |x_j - y_j|^p)^{1/p} = ||\mathrm{x} - \mathrm{y}||_p$$

The *2-norm* refers to the familiar *Euclidean distance*

$$Dis_2(x,y) = \sqrt{\sum_{j=1}^{n} (x_j - y_j)^2} = \sqrt{(x-y)^T(x-y)}$$

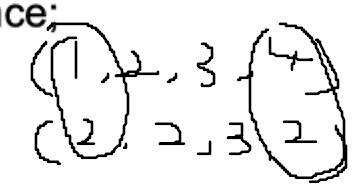
The *1-norm* denotes *Manhattan distance*, also called *cityblock* distance:

$$Dis_1(x,y) = \sum_{j=1}^{n} |x_j - y_j|$$

Comp9417
Distance

- If we now let p grow larger, the distance will be more and more dominated by the largest coordinate-wise distance, from which we can infer that $Dis_\infty = max_j|x_j - y_j|$; this is also called *Chebyshev distance*.
- You will sometimes see references to the *0-norm* (or L_0 norm) which counts the number of non-zero elements in a vector. The corresponding distance then counts the number of positions in which vectors x and y differ. This is not strictly a Minkowski distance; however, we can define it as:

$$Dis_0(x,y) = \sum_{j=1}^{d}(x_j - y_j)^0 = \sum_{j=1}^{d} I[x_j \neq y_j]$$



几个不同 distance就等于几

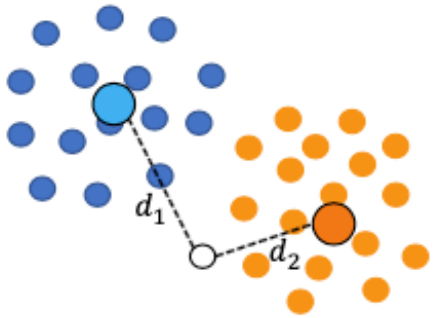
- If x and y are binary strings, this is also called the *Hamming distance*. Alternatively, we can see the Hamming distance as the number of bits that need to be flipped to change x into y.

Comp9417

Nearest Centroid Classifier

Nearest centroid classifier

- Distance based classifier
- $\mu_k = \frac{1}{|C_k|}\sum_{j \in C_k} x_j$
- For complex classes (eg. Multimodal, non-spherical) may give very poor results
- Can not handle outliers and noisy data well
- Not very accurate



Comp9417

Nearest Neighbor

Stores all training examples $\langle x_j, f(x_j) \rangle$.

Nearest neighbour:

- Given query instance x_q, first locate nearest training example x_n, then estimate $\hat{f}(x_q) \leftarrow f(x_n)$

k-Nearest neighbour:

- Given x_q, take vote among its k nearest neighbours (if discrete-valued target function) (see next slide)
- take mean of f values of k nearest neighbours (if real-valued)

$$\hat{f}(x_q) \leftarrow \frac{\sum_{j=1}^{k} f(x_j)}{k}$$

Comp9417

Nearest Neighbor

- Distance base classifier
- Find k nearest neighbor using an appropriate distance metric (e.g. Minkowski distance)
- Predict the output based on the majority vote
- Works better with lots of training data and small number of attributes
- Can be very accurate but slow at testing time
- Curse of dimensionality
- Assumes all attributes are equally important
    - Remedy: attribute selection or attribute weights
- Needs homogenous feature type and scale

- kNN uses the training data as exemplars, so using simple search for prediction is $O(n)$!

Comp9417

Distance-Weighted KNN

- Might want to weight nearer neighbours more heavily ...
- Use distance function to construct a weight w_i
- Replace the final line of the classification algorithm by:

$$\hat{f}(x_q) \leftarrow \arg\max_{v \in V} \sum_{i=1}^{k} w_i \delta(v, f(x_j)) \qquad \hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

Where

$$w_i = \frac{1}{Dis(x_q, x_i)^2}$$

$Dis(x_q, x_i)$ is distance between x_q, x_i

Comp9417
Evaluation

Lazy learners do not construct an explicit model, so how do we evaluate the output of the learning process ?

- 1-NN – training set error is always zero !
  - each training example is always closest to itself
- k-NN – overfitting may be hard to detect

Solution:

Leave-one-out cross-validation (LOOCV) – leave out each example and predict it given the rest:

$$(x_1, y_1), (x_2, y_2), \ldots, (x_{i-1}, y_{i-1}), (x_{i+1}, y_{i+1}), \ldots, (x_m, y_m)$$

Error is mean over all predicted examples. Fast – no models to be built !

Comp9417

Sample

Consider the following truth table which gives an "m–of–n function" for three Boolean variables, where "1" denotes true and "0" denotes false. In this case the target function is: "exactly two out of three variables are true".

X	Y	Z	Class
0	0	0	false
0	0	1	false
0	1	0	false
0	1	1	true
1	0	0	false
1	0	1	true
1	1	0	true
1	1	1	false

Suppose we define a simple measure of distance between two equal length strings of Boolean values, as follows. The distance between two such strings B_1 and B_2 is:

$$\text{distance}(B_1, B_2) = |(\sum B_1) - (\sum B_2)|$$

where $\sum B_i$ is simply the number of variables with value 1 in string B_i. For example:

$$\text{distance}(\langle 0,0,0\rangle, \langle 1,1,1\rangle) = |0 - 3| = 3$$

and

$$\text{distance}(\langle 1,0,0\rangle, \langle 0,1,0\rangle) = |1 - 1| = 0$$

What is the LOOCV ("Leave–one–out cross-validation") error of 2–Nearest Neighbour using our distance function on the examples in the table ? [Show your working.]

Comp9417

Inductive Bias

"Inductive bias" is the combination of assumptions and restrictions placed on the models and algorithms used to solve a learning problem.

For example, what is the inductive bias of:

- Linear Regression ?
  - target function has the form $y = ax + b$
  - approximate by fitting using MSE
- Nearest Neighbour ?
  - target function is a complex non-linear function of the data
  - predict using nearest neighbour by Euclidean distance in feature space

Comp9417
Bayes Theorem

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

where:

$P(h)$ = prior probability of hypothesis h

$P(D)$ = prior probability of training data D

$P(h|D)$ = probability of h given D

$P(D|h)$ = probability of D given h

If the output belongs to a set of k classes: $y \in \{C_1, C_2, \ldots, C_k\}$ for $1 \leq i \leq k$

Then in Bayesian framework:

MAP $P(y = C_i|x) = \frac{P(x|C_i)P(C_i)}{P(x)}$ ML

- $P(y = C_i|x)$: posterior probability
- $P(x|C_i)$: class conditional (likelihood)
- $P(C_i)$: prior
- $P(x)$: Marginal ($P(x) = \sum_i p(x|C_i).P(C_i)$)

$\begin{cases} P(y=0|x) \\ P(y=1|x) \end{cases}$ 大的

$\begin{cases} P(x|y=0) \\ P(x|y=1) \end{cases}$ 大的

Comp9417

Bayes Theorem

Generally want the most probable hypothesis given the training data,
Maximum a posteriori hypothesis h_{MAP} :

$$h_{MAP} = \arg\max_{h \in H} P(h|D)$$

$$= \arg\max_{h \in H} \frac{P(D|h)P(h)}{P(D)}$$

$$= \arg\max_{h \in H} P(D|h)P(h)$$

If assume $P(h_i) = P(h_j)$ then can further simplify, and choose the *Maximum likelihood* (ML) hypothesis:

$$h_{ML} = \arg\max_{h_i \in H} P(D|h_i)$$

Comp9417

Bayesian Methods

Example: Imagine, we want to classify fish type: Salmon , Sea bass

If from the past experience we have (C_i *is the class*):

$P(c_i)$	Salmon	Sea bass
Prior	0.3	0.7

- If we decide only based on prior, we always have to choose "sea bass". This is called "*decision rule based on prior*"
  - This can behave vey poorly
  - It never predicts other classes

Comp9417
Bayes Theorem

Example: now if we have some more information on the length of the fish in each class, then how can we update our decision, if we want to predict the class for a fish with 70cm length?

$P(x\|c_i)$	Salmon	Sea bass
length > 100 cm	0.5	0.3
50 cm < length < 100 cm	0.4	0.5
length < 50 cm	0.1	0.2

$$P(c = salmon|x = 70cm) \propto P(70cm|salmon) * P(salmon) = 0.4*0.3=0.12$$
$$P(c = sea\ bass|x = 70cm) \propto P(70cm|sea\ bass) * P(sea\ bass) = 0.5*0.7=0.35$$

So base on these probabilities, our model predict the type as “sea bass”

Comp9417

Naïve Bayes Classifier

Assume target function $f : X \rightarrow V$, where each instance x described by attributes $\langle x_1, x_2, \ldots, x_n \rangle$.
Most probable value of $f(x)$ is:

$$v_{MAP} = \arg\max_{v_j \in V} P(v_j | x_1, x_2, \ldots, x_n)$$

$$v_{MAP} = \arg\max_{v_j \in V} \frac{P(x_1, x_2, \ldots, x_n | v_j) P(v_j)}{P(x_1, x_2, \ldots, x_n)}$$

$$= \arg\max_{v_j \in V} P(x_1, x_2, \ldots, x_n | v_j) P(v_j)$$

Comp9417

Naïve Bayes Classifier

Naive Bayes assumption:

$$P(x_1, x_2, \ldots, x_n | v_j) = \prod_i P(x_i | v_j)$$

- Attributes are statistically independent (given the class value)
  - which means knowledge about the value of a particular attribute tells us nothing about the value of another attribute (if the class is known)

Which gives **Naive Bayes classifier:**

$$v_{NB} = arg \max_{v_j \in V} P(v_j) \prod_i P(x_i | v_j)$$

Comp9417

Naïve Bayes Classifier

Naive Bayes Learn (examples)

for each target value v_j

$\hat{P}(v_j) \leftarrow$ estimate $P(v_j)$

For each attribute value x_i:

$\hat{P}(x_i|v_j) \leftarrow$ estimate $P(x_i|v_j)$

Classify New Instance (for sample x)

$$v_{NB} = arg \max_{v_j \in V} \hat{P}(v_j) \prod_i \hat{P}(x_i|v_j)$$

Comp9417
Example

outlook	temperature	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

What are the required probabilities to predict *PlayTennis*?

Outlook			Temperature			Humidity			Windy		
	Yes	No		Yes	No		Yes	No		Yes	No
Sunny	2	3	Hot	2	2	High	3	4	False	6	2
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3
Rainy	3	2	Cool	3	1						
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5
Rainy	3/9	2/5	Cool	3/9	1/5						

Play	
Yes	No
9	5
9/14	5/14

Comp9417
Example

Say we have the new instance:

$$\langle Outlk = sun,\ Temp = cool,\ Humid = high,\ Wind = true \rangle$$

We want to compute:

$$v_{NB} = arg \max_{v_j \in \{\text{“yes”},\text{“no”}\}} P(v_j) \prod_i P(x_i|v_j)$$

So we first calculate the likelihood of the two classes, “yes” and “no”

For “yes” $= P(\text{“yes”}) \times P(sun|\text{“yes”}) \times P(cool|\text{“yes”}) P(high|\text{“yes”}) \times P(true|\text{“yes”})$

$$0.0053 = \frac{9}{14} \times \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9}$$

For “no” $= P(\text{“no”}) \times P(sun|\text{“no”}) \times P(cool|\text{“no”}) P(high|\text{“no”}) \times P(true|\text{“no”})$

$$0.0206 = \frac{4}{14} \times \frac{3}{5} \times \frac{1}{5} \times \frac{4}{9} \times \frac{3}{9}$$

Then convert to a probability by normalisation

$$P(\text{“yes”}) = \frac{0.0053}{(0.0053 + 0.0206)} = 0.205$$

$$P(\text{“no”}) = \frac{0.0206}{(0.0053 + 0.0206)} = 0.795$$

The Naive Bayes classification is “no”.

Comp9417

Zero frequency

What if none of the training instances with target value v_j have attribute value x_i? Then

$$\hat{P}(x_i|v_j) = 0, and \ ...$$

$$\hat{P}(v_j)\prod_i \hat{P}(x_i|v_j) = 0$$

Pseudo-counts add 1 to each count (a version of the *Laplace Estimator*)

(In some cases adding a constant different from 1 might be more appropriate)

Comp9417
Numeric attributes

- Usual assumption: attributes have a *normal* or *Gaussian* probability distribution (given the class)
- The *probability density function* for the normal distribution is define by two parameters:
  - The sample mean μ:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

  - The standard deviation σ:

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

These parameters have to be defined for each class separately.

Then we have the density function $f(x)$:

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

Example: continuous attribute ***temperature*** with $mean = 73$ and $standard\ deviation = 6.2$. Density value

$$f(temprature = 66|"yes") = \frac{1}{\sqrt{2\pi}6.2} e^{\frac{(66-73)^2}{2\times 6.2^2}} = 0.0340$$

Missing values during training are not included in calculation of mean and standard deviation.

Comp9417

Text Classification

Naive Bayes conditional independence assumption

$$P(doc|v_j) = \prod_{i=1}^{length(doc)} P(x_i = w_k | v_j)$$

where $P(x_i = w_k|v_j)$ is probability that word in position i is w_k, given v_j

one more assumption:

$$P(x_i = w_k|v_j) = P(x_m = w_k|v_j), \forall i, m$$

“bag of words”

Comp9417
Text Classification

We again use the example of Naive Bayes for text classification to illustrate, using both the multinomial and multivariate Bernoulli models.

Consider the following e-mails consisting of five words a, b, c, d, e:

$e1$: $b\ d\ e\ b\ b\ d\ e$	$e5$: $a\ b\ a\ b\ a\ b\ a\ e\ d$
$e2$: $b\ c\ e\ b\ b\ d\ d\ e\ c\ c$	$e6$: $a\ c\ a\ c\ a\ c\ a\ e\ d$
$e3$: $a\ d\ a\ d\ e\ a\ e\ e$	$e7$: $e\ a\ e\ d\ a\ e\ a$
$e4$: $b\ a\ d\ b\ e\ d\ a\ b$	$e8$: $d\ e\ d\ e\ d$

We are told that the e-mails on the left are spam and those on the right are ham, and so we use them as a small training set to train our Bayesian classifier.

- First, we decide that d and e are so-called stop words that are too common to convey class information.
- The remaining words, a, b and c, constitute our vocabulary.

Comp9417
Multivariate Bernoulli

E-mail	$\#a$	$\#b$	$\#c$	Class
e_1	0	1	0	+
e_2	0	3	1	+
e_3	3	0	0	+
e_4	1	3	0	+
e_5	4	3	0	−
e_6	1	0	1	−
e_7	3	0	0	−
e_8	0	0	0	−

In the multivariate Bernoulli model e-mails are represented by bit vectors, as before.

- Adding the bit vectors for each class results in $(2, 3, 1)$ for spam and $(3, 1, 1)$ for ham.
- Each count is to be divided by the number of documents in a class, in order to get an estimate of the probability of a document containing a particular vocabulary word.
- Probability smoothing now means adding two pseudo-documents, one containing each word and one containing none of them.
- This results in the estimated parameter vectors

$$\hat{\theta}^{\oplus} = \left(\frac{3}{6}, \frac{4}{6}, \frac{2}{6}\right) = (0.5, 0.67, 0.33) \quad \text{for spam}$$

$$\hat{\theta}^{\ominus} = \left(\frac{4}{6}, \frac{2}{6}, \frac{2}{6}\right) = (0.67, 0.33, 0.33) \quad \text{for ham}$$

Comp9417
Multivariate Bernoulli

Suppose our vocabulary contains three words a, b and c, and we use a multivariate Bernoulli model for our e-mails, with parameters

$$\theta^{\oplus} = (0.5,0.67,0.33) \qquad \theta^{\ominus} = (0.67,0.33,0.33)$$

This means, for example, that the presence of b is twice as likely in spam (+), compared with ham (−).

The e-mail to be classified contains words a and b but not c, and hence is described by the bit vector $x = (1,1,0)$. We obtain likelihoods

$$P(x|\oplus) = 0.50\times0.67\times(1-0.33) = 0.222$$

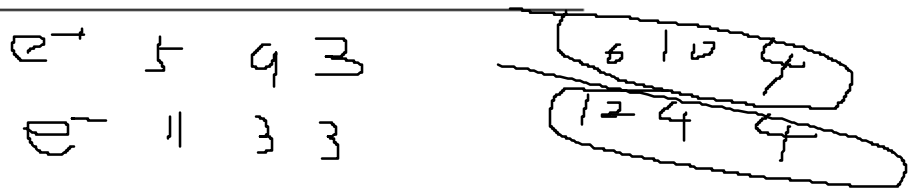
$$P(x|\ominus) = 0.67\times0.33\times(1-0.33) = 0.148$$

← spam ⊕

The ML classification of x is thus spam.

Comp9417
Multinomial

E-mail	$\#a$	$\#b$	$\#c$	Class
e_1	0	3	0	+
e_2	0	3	3	+
e_3	3	0	0	+
e_4	2	3	0	+
e_5	4	3	0	−
e_6	4	0	3	−
e_7	3	0	0	−
e_8	0	0	0	−



For the multinomial model, we represent each e-mail as a count vector, as before.

- In order to estimate the parameters of the multinomial, we sum up the count vectors for each class, which gives $(5, 9, 3)$ for spam and $(11, 3, 3)$ for ham.
- To smooth these probability estimates we add one pseudo-count for each vocabulary word, which brings the total number of occurrences of vocabulary words to 20 for each class.
- The estimated parameter vectors are thus

$$\hat{\theta}^{\oplus} = \left(\frac{6}{20}, \frac{10}{20}, \frac{4}{10}\right) = (0.3, 0.5, 0.2) \quad \text{for spam}$$

$$\hat{\theta}^{\ominus} = \left(\frac{12}{20}, \frac{4}{20}, \frac{4}{10}\right) = (0.6, 0.2, 0.2) \quad \text{for ham}$$

Comp9417
Multinomial

Alternatively, we can employ a multinomial model. The parameters of a multinomial establish a distribution over the words in the vocabulary, say

$$\theta^{\oplus} = (0.3,0.5,0.2) \qquad \theta^{\ominus} = (0.6,0.2,0.2)$$

The e-mail to be classified contains three occurrences of word a, one single occurrence of word b and no occurrences of word c, and hence is described by the count vector $x = (3,1,0)$. The total number of vocabulary word occurrences is $n = 4$. We obtain likelihoods:

$$P(x|\oplus) = 4!\frac{0.3^3}{3!}\frac{0.5^1}{1!}\frac{0.2^0}{0!} = 0.054$$

$$P(x|\ominus) = 4!\frac{0.6^3}{3!}\frac{0.2^1}{1!}\frac{0.2^0}{0!} = 0.1728$$

The likelihood ratio is $(\frac{0.3}{0.6})^3(\frac{0.5}{0.2})^1(\frac{0.2}{0.2})^0 = \frac{5}{16}$. The ML classification of x is thus ham, the opposite of the multivariate Bernoulli model. This is mainly because of the three occurrences of word a, which provide strong evidence for ham.

Comp9417
Logistic Regression

- We can write the *posterior odds*:

$$\frac{P(Y=1|x)}{P(Y=0|x)} = \frac{P(x|Y=1)}{P(x|Y=0)}\frac{P(Y=1)}{P(Y=0)}$$

- If we take the $log_e\ (ln)$ then:

$$ln\frac{P(Y=1|x)}{P(Y=0|x)} = ln\frac{P(x|Y=1)}{P(x|Y=0)} + ln\frac{P(Y=1)}{P(Y=0)}$$

- *Simplifying assumption:* We assume the log likelihood ratio is a linear function of x

$$ln\frac{P(x|Y=1)}{P(x|Y=0)} = x\beta$$

- $ln\frac{P(Y=1)}{P(Y=0)}$ is a fixed value, so we can define $\beta_0 = ln\frac{P(Y=1)}{P(Y=0)}$
- So, we have:

$$ln\frac{P(Y=1|x)}{P(Y=0|x)} = x\beta + \beta_0$$

Now we have a linear solution to our problem and this is what makes *Logistic Regression* a linear model.

Comp9417
Logistic Regression

- How to get our probability values:

$$ln\frac{P(Y=1|x)}{1-P(Y=1|x)} = x\beta + \beta_0$$

$$\frac{P(Y=1|x)}{1-P(Y=1|x)} = e^{x\beta+\beta_0}$$

$$P(Y=1|D) = \frac{e^{x\beta+\beta_0}}{1+e^{x\beta+\beta_0}} = \frac{1}{1+e^{-(x\beta+\beta_0)}}$$

Generalises to multiple class versions (Y can have more than two values).