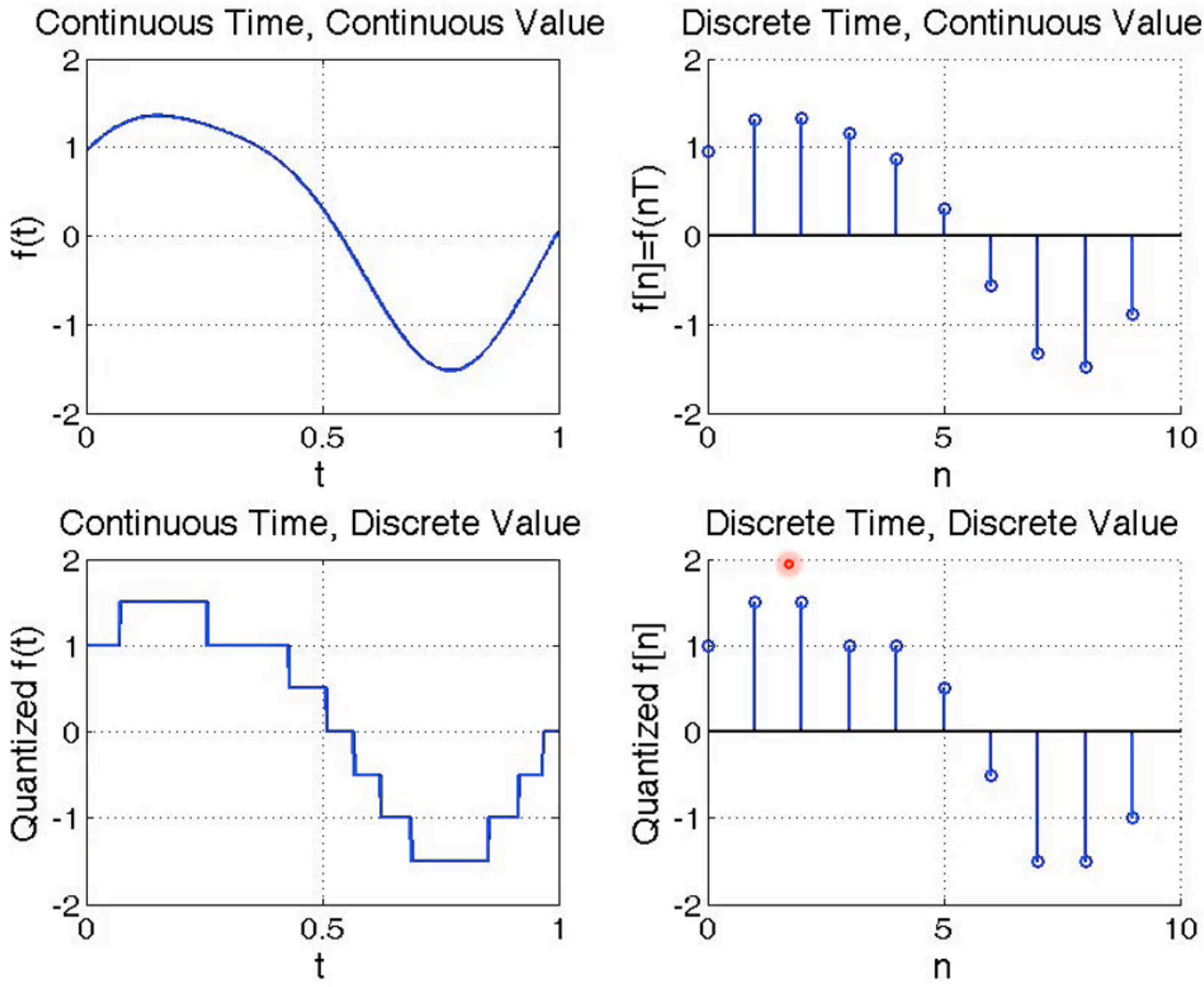
Week 13

1. Sample and quantise



2. Wave period and frequency

f = 1 / T

3. Nyquist sampling rate

Sampling rate s should be at least twice the highest spatial frequency u.

s >= 2u

4. Distance measurent

Distance is measure of separation between data.

A valid distance measure D(a,b) has the following **properties:**

1. Non-negative D(a,b) >= 0
2. Reflexive: D(a,b) = 0 <==> a = b
3. Symmetric: D(a,b) = D(b,a)
4. Satisfies triangular inequality: D(a,b) <= D(a,c) + D(c,b)

- Minkowski distance

$$D(x,y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}}$$

- Manhattan distance aka 1-norm distance

$$D(x,y) = \sum_{i=1}^{n} |x_i - y_i|$$

- **Euclidean distance** aka 2-norm distance

$$D(x, y) = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}$$

Can be expressed in vector form:

$$D(x, y) = \| \mathbf{x} - \mathbf{y} \|$$

$$= \sqrt{(\mathbf{x} - \mathbf{y})^T (\mathbf{x} - \mathbf{y})}$$

- Chebyshev distance aka p = infinity

$$D(x,y) = \lim_{p\to\infty}\left(\sum_{i=1}^{n}|x_i - y_i|^p\right)^{\frac{1}{p}}$$

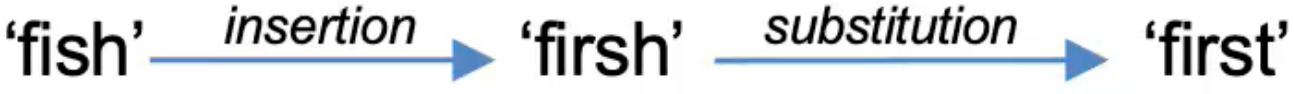
$$= max\left(|x_1 - y_1|, |x_2 - y_2|, \ldots, |x_n - y_n|\right)$$

- Hamming distance (distance between **symbolic data**)

1. Defined over symbolic data of the **same length**
2. Measures the number of **substitutions** required to change not string/number into another

- Edit distance

1. Defined on text data of **any length**
2. Measures the minimum number of 'operations' required one sequence of characters into another.
3. 'Operations' can be: **insertion, substitution, deletion**



5. Dynamic Time Warping(Numerical Time Series)

- Replaces Euclidean one-to-one comparison with many-to-one

DTW can be defined recursively:

For two time series $\boldsymbol{X} = (x_0, .., x_n)$ and $\boldsymbol{Y} = (y_0, .., y_m)$

$$DTW(\boldsymbol{X}, \boldsymbol{Y}) = D(x_0, y_0) + \min\{DTW(\boldsymbol{X}, REST(\boldsymbol{Y})), DTW(REST(\boldsymbol{X}), \boldsymbol{Y}), DTW(REST(\boldsymbol{X}), REST(\boldsymbol{Y}))\}$$

where $REST(\boldsymbol{X}) = (x_1, .., x_n)$

6. Covariance matrix

In three dimensions,

$$\mathbf{C} = \frac{1}{N}\sum_i \begin{bmatrix} (v_{i1}-\mu_1)^2 & (v_{i1}-\mu_1)(v_{i2}-\mu_2) & (v_{i1}-\mu_1)(v_{i3}-\mu_3) \\ (v_{i1}-\mu_1)(v_{i2}-\mu_2) & (v_{i2}-\mu_2)^2 & (v_{i2}-\mu_2)(v_{i3}-\mu_3) \\ (v_{i1}-\mu_1)(v_{i3}-\mu_3) & (v_{i2}-\mu_2)(v_{i3}-\mu_3) & (v_{i3}-\mu_3)^2 \end{bmatrix}$$

A Covariance matrix is always:

- square and symmetric
- variances on the diagonal
- covariance between each pair of dimensions is included in non-diagonal elements

7. Eigen analysis

- Eigenvectors and eigenvalues define **principal axes** and spread of points along directions
- **Major axis** - eigenvector corresponding to **larger eigenvalue**(i.e. **larger variance**)
- **Minor axis** - eigenvector corresponding to **smaller eigenvalu**e(i.e. **smaller variance)**

To calculate eigenvectors of a square matrix e.g. a covariance matrix**(C)**, then solve

ve

$$|\mathbf{C} - \lambda \boldsymbol{I}| = 0$$

where

- $\boldsymbol{I}$ is the identity matrix
- $|\mathbf{C}|$ is the determinant of the matrix

|C| = ad - bc

λ is the **eigenvalue**.

To get the **eigenvector v** : Cv = λv

8. Mean and Variance and Standard Deviation

For one-dimensional data $\mathbf{v} = \{v_1, .., v_n\}$,

Mean: [average] $$\mu = \frac{1}{N} \sum_i v_i$$

Variance: [spread] $$\sigma^2 = \frac{1}{N-1} \sum_i (v_i - \mu)^2$$

Standard Deviation: $$\sigma = \sqrt{\frac{1}{N-1} \sum_i (v_i - \mu)^2}$$

9. Gaussian Distribution

One dimension:

For a normal distribution $N(\mu, \sigma^2)$ in one dimension, the probability density function (pdf) can be calculated as:

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

Multi-dimension:

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

WARNING: **Σ is the capital letter of σ, not the summation sign! So here Σ is the covariance matrix.**

10. Data normalisation

Methods for normalisation:

1. Rescaling $$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

2. Standardisation (also known as z-score) $$x' = \frac{x - \mu}{\sigma}$$

3. Scaling to unit length $$x' = \frac{x}{||x||}$$

Week 14

1. Model

Model == a structured approach for making predictions about the future.

2.Maximum likelihood fitting of a Bernoulli

$$P(x|p) = \text{Bernoulli}\,(x;p)$$

$$\text{Bernoulli}\,(x;p) = \begin{cases} 1-p & \text{if } x = 0 \\ p & \text{if } x = 1 \end{cases}$$

The maximum-likelihood probability is the empirical **mean** of the data points. (This works well for a reasonable number of datapoints.)

$$\mathcal{L}(p) = \sum_\lambda \log \mathrm{P}\,(x_\lambda|p)$$

$$\mathcal{L}(p) = \left(\sum_\lambda x_\lambda\right) \log p + \left(N - \sum_\lambda x_\lambda\right) \log(1-p)$$

$$\begin{aligned} 0 &= \left.\frac{\partial \mathcal{L}(p)}{\partial p}\right|_{p=\hat{p}} \\ 0 &= \frac{\sum_\lambda x_\lambda}{\hat{p}} - \frac{N - \sum_\lambda x_\lambda}{1-\hat{p}} \\ \frac{\sum_\lambda x_\lambda}{\hat{p}} &= \frac{N - \sum_\lambda x_\lambda}{1-\hat{p}} \\ (1-\hat{p}) \sum_\lambda x_\lambda &= \hat{p}\left(N - \sum_\lambda x_\lambda\right) \\ \sum_\lambda x_\lambda &= \hat{p} N \\ \hat{p} &= \tfrac{1}{N} \sum_i x_i \end{aligned}$$

3. Bayesian fitting of a Bernoulli

(If we only have very little data)

The law of **joint probability**:

$$\mathrm{P}(\mathbf{x}, p) = \mathrm{P}(\mathbf{x}|p)\mathrm{P}(p) = \mathrm{P}(p|\mathbf{x})\mathrm{P}(\mathbf{x})$$

$$\mathrm{P}(p|\mathbf{x}) = \frac{\mathrm{P}(\mathbf{x}|p)\mathrm{P}(p)}{\mathrm{P}(\mathbf{x})} \propto \mathrm{P}(\mathbf{x}|p)\mathrm{P}(p)$$

$$\log \mathrm{P}(p|\mathbf{x}) = \mathcal{L}(p) + \log \mathrm{P}(p) + \text{const}$$

3.a MAP (maximum a posterior)

Given the value of y, what value of x is most likely?

P(x|y)

3.b Maximum Likelihood estimate

$$Posterior = \frac{Likelihood \times Prior}{Evidence}$$

Maximum likelihood estimate uses the **likelihood** only, P(D|θ), to calculate an estimate for θ.

MAP builds on this by also introducing **prior.**

4. Supervised learning: Regression and classification

The inputs and outputs could be anything(image, audio)

The goal is to learning something about the mapping from x to y

Control + command + space = emoji and symbols

To write a function that takes an X and returns a distribution over y

f(x) -> P(y|x)

Classification and Regression are 2 special types of supervised learning.

Classification : the output is a class-label

Regression : the output is one (or many) real values

Week 15

0. Multivariate normal distribution

Normal Gaussian PDF :

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

What if x is not one-dimensional?

Multivariate Gaussian distribution PDF:

$$P(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{-1/2}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}$$

Σ is the covariance matrix

1. Regression

If you don't want to give up on using the more complex basis function to capture non-linearities

We can penalise the weights

$$\mathcal{L}(\mathbf{w}) = \log \mathrm{P}(\mathbf{y}|\mathbf{X}, \mathbf{w}) - \tfrac{1}{2}\mathbf{w}^T \mathbf{\Lambda} \mathbf{w}$$

when $\mathbf{\Lambda} = 0$, out estimate will be the same as our log-likelihood estimate.

The aim is to find the weights $\mathbf{w}$:

We are trying to estimate $\boldsymbol{w}$, let's pretend that it has 2-dimensions and hence $\boldsymbol{w} = (w_1\ w_2)$.

$$-\frac{1}{2}\boldsymbol{w}^T \boldsymbol{\Lambda} \boldsymbol{w} = -\frac{1}{2} \cdot \begin{pmatrix} w_1 & w_2 \end{pmatrix} \cdot \begin{pmatrix} \Lambda_{11} & 0 \\ 0 & \Lambda_{22} \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \tag{1}$$

$$= -\frac{1}{2} \cdot \begin{pmatrix} w_1 & w_2 \end{pmatrix} \cdot \begin{pmatrix} \Lambda_{11} w_1 \\ \Lambda_{22} w_2 \end{pmatrix} \tag{2}$$

$$= -\frac{1}{2}(\Lambda_{11} w_1^2 + \Lambda_{22} w_2^2) \tag{3}$$

The derivate of the first part:

$\log \mathbf{P}(\mathrm{y}|\mathbf{X}, \mathbf{w}) =$ $\frac{1}{\sigma^2} \mathbf{X}^T (\mathbf{y} - \mathbf{X}\mathbf{w})$

2. Overfitting(sometimes regression can fail)

- Too little data
- Too little data in some directions
- The function class is too complex

---

how to mitigate overfitting:

3. Cross-validation

we split the data into "training" and "validation" sets. We train the model on the training set, then look at the residuals/errors on the validation set.

The model with the smallest cross-validation error wins!

4. Regularisation

If the data is very complex and may need to use complex functions which might also cause overfitting

If you don't want to give up on using the more complex basis functions to capture non-linearities.

We can penalise the weights

$$\mathcal{L}(\mathbf{w}) = \log \mathrm{P}\left(\mathbf{y}|\mathbf{X}, \mathbf{w}\right) - \tfrac{1}{2}\mathbf{w}^T\mathbf{\Lambda}\mathbf{w}$$

where we could (if we wanted) **penalise** different weights differently, using the positive **diagonal matrix**, Λ.(its a lambda)

5. Limits of cross-validation

Cross-validation in combination with regularisation can be very helpful

But :

- Parameter sweeps can be numerically costly.
- Splitting your data gives you less data for training which is very problematic with smaller amounts of data.
- Scales poorly if you want to cross-validate many different parameters.

Week 16

1. Classification

Classification is almost exactly the same as regression, except that:

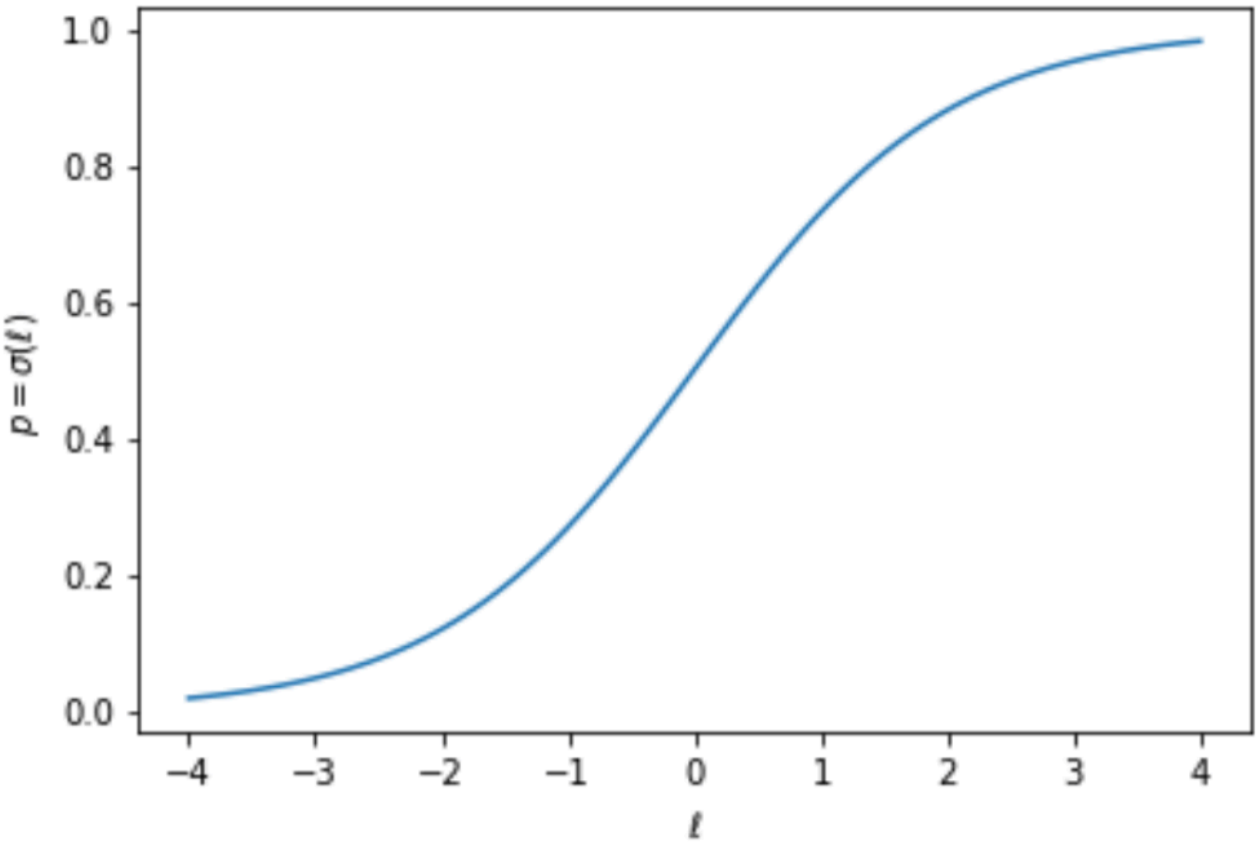
- The outputs, y, are discrete class-labels.
- Almost all interesting/useful algorithms require iterative solutions.

2. Login parameterisation of Bernoulli distribution

- Probabilities live in a strange range, [0,1]
- Most of the algorithms output between -∞ to ∞.

Instead we can also treat the Bernoulli parameter as a logits vector, l, defined such that,

Sigmoid function :



$$p = \text{sigmoid}(\ell) = \sigma(\ell)$$

$$p = \frac{1}{1 + e^{-\ell}}$$

3. Categorical distribution for multi-class classification

Samples from the Categorical distribution are integers 0⩽y<K, with probability given explicity by a length K vector of, p,

Samples from the Categorical distribution are integers $0 \leq y < K$, with probability given explicity by a length K vector of, $\mathbf{p}$,

$$\mathrm{P}(y|\mathbf{p}) = \mathrm{Categorical}(y; \mathbf{p}) = p_y$$

Softmax:

- Probabilities live in a strange range, [0,1]
- Most of the algorithms output between -∞ to ∞.
- **Probabilities must sum to 1**

To make it always sum to 1, we can also treat the Categorical parameter as a logits vector, ℓ, defined such that,

$$\mathbf{p} = \operatorname{softmax}(\boldsymbol{\ell})$$

$$p_i = \frac{e^{\ell_i}}{\sum_{j=0}^{K-1} e^{\ell_j}}$$

Other simpler, heuristic methods for classification:

5. K-nearest neighbour

Look at the nearby datapoints. If the nearby points come from one class, then the chances are that our datapoint come from the same class.

6. Weighted-nearest neighbour

It soften the boundaries than k-nearest neighbour

$$p_y(\mathbf{x}) = \frac{\sum_\lambda k(\mathbf{x}, \mathbf{x}_\lambda)\delta_{y,y_\lambda}}{\sum_\lambda k(\mathbf{x}, \mathbf{x}_\lambda)}$$

δ chronic delta : 1 if y and yλ are the same , 0 otherwise

$$k(\mathbf{x}_\lambda, \mathbf{x}_{\lambda'}) = e^{-(\mathbf{x}_\lambda - \mathbf{x}_{\lambda'})^2/(2b)}$$

Where b is a bandwidth parameter.

7. Nearest centroids

Here we compute the centre of the inputs for each class(mean), then we're classifying the points based on the centroid which is the closest.

But the class boundary tends to be diagonal to the centroids line. **Nearest centroid works quite poorly, because it only takes into account the mean of the classes, not their shape**

8. Bayes theorem

Use Bayes theorem to take the shape into account

Here, we separately learn a distribution over data-points for a single class, P(x|y).

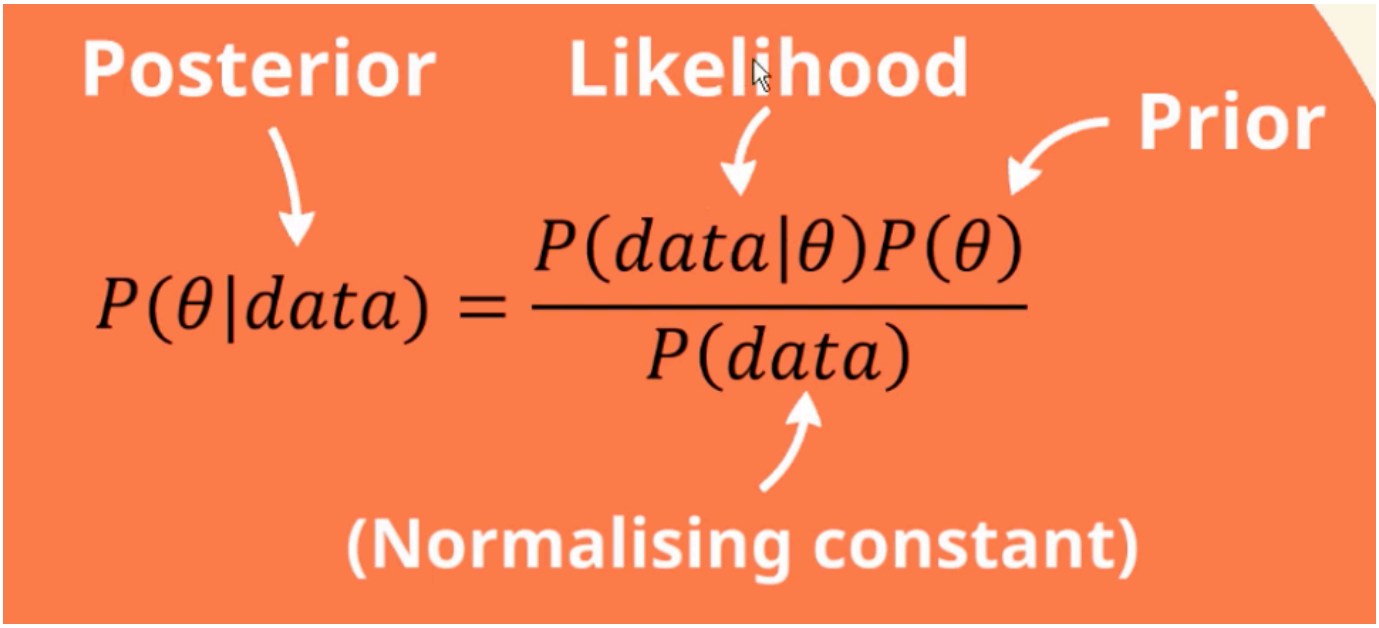
In particular, we can fit multivariate normals to the input points,

$$\mathrm{P}(\mathbf{x}|y) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y).$$

Bayes Rule:

θ : the parameter data : the data sample

P(data): in order to make the whole thing sums to one(because its probability distribution!!!!)



Since P(data) has nothing to do with θ, we can also say that

$$P(\theta|data) \propto P(data|\theta)P(\theta)$$

Week 17

1. Bayes classification scheme:

use Bayes theorem to give us the probability of a class, conditioned on a data-point,

$$\mathrm{P}(y|\mathbf{x}) = \frac{\mathrm{P}(y)\mathrm{P}(\mathbf{x}|y)}{\mathrm{P}(\mathbf{x})} \propto \mathrm{P}(y)\mathrm{P}(\mathbf{x}|y).$$

$$\mathrm{P}(\mathbf{x}|y) = \mathcal{N}\left(\mathbf{x}; \boldsymbol{\mu}_y, \sigma^2\mathbf{I}\right).$$

$$\mathrm{P}(y) = 1/K$$

$$\mathrm{P}(y|\mathbf{x}) \propto \mathrm{P}(\mathbf{x}|y) = \mathcal{N}\left(\mathbf{x}; \boldsymbol{\mu}_y, \sigma^2\mathbf{I}\right).$$

Critically, once we have this probabilistic representation, we get a recipe for doing a better job, by taking into account the **shape** of the input distributions. In particular, we can fit **multivariate normals** to the input points,

$$\mathrm{P}(\mathbf{x}|y) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y).$$

2. Naive Bayes

$$P(\mathbf{x}_\lambda|y_\lambda) = \prod_i P(x_{\lambda,i}|y_\lambda)$$

The probability of an email, xλ, is a product over the probability of each word

However, before we can fully reach that conclusion, we need to incorporate the prior and renormalise.

$$\mathrm{P}(y_\lambda|\mathbf{x}_\lambda) \propto \mathrm{P}(y_\lambda)\mathrm{P}(\mathbf{x}_\lambda|y_\lambda)$$

```
propto_post_spam    = 0.8*like_spam
propto_post_notspam = 0.2*like_notspam

(propto_post_spam, propto_post_notspam)
```

```
(0.0864, 0.00015000000000000001)
```

```
post_spam    = propto_post_spam    / (propto_post_spam + propto_post_notspam)
post_notspam = propto_post_notspam / (propto_post_spam + propto_post_notspam)

(post_spam, post_notspam)
```

```
(0.9982668977469671, 0.001733102253032929)
```

Unsupervised learning and clustering:

2. Clustering

Clustering is not classification

and unsupervised learning is not supervised learning

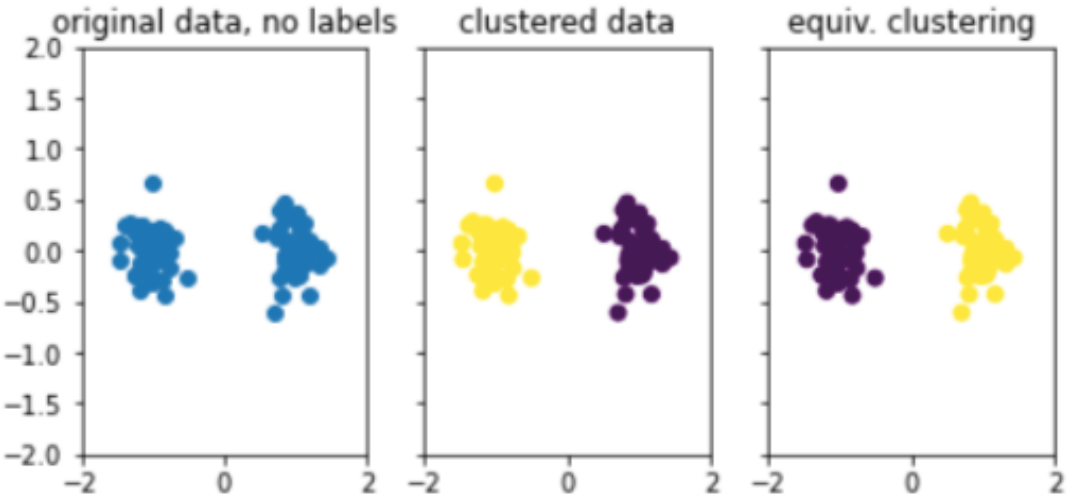
```
#### Supervised learning
x : X # Input
y : Y # Output
    [(X, Y)] -> (X -> Y)
```

In supervised learning, the goal is to lean a function that maps from a new input test point, to a distribution over the corresponding y.

```
#### Unsupervised learning
x : X # Input
z : Z # Latent
    [X] -> [Z]
```

Give a list of inputs and the goal is to compute a latent variable that somehow summarises the **structure** of the inputs.

All data below are 2D inputs x.



For some problem, we might need to **ignore clusters.**

3. Cluster centers, Coordinate descent, K-means clustering

Cluster centers: ($\boldsymbol{\mu}$ is a vector)

$$\boldsymbol{\mu} = \text{cluster center, for cluster indexed } z$$

integer-valued cluster assignments, zi, describing the cluster associated with the ith datapoint,:

$$z_i = \text{integer cluster index for each datapoint, indexed } i$$

The objective is to find a bunch of cluster centers such that all datapoints are close to a cluster center. **We minimise the squared distance between each datapoint and its assigned cluster centre.**

$$\mathcal{L}(\mathbf{z}, \boldsymbol{\mu}) = \sum_i \left\| \mathbf{x}_i - \boldsymbol{\mu}_{z_i} \right\|^2$$

Coordinate descent (alternate between optimising z and μ):

1. First, we assign each datapoint to the **nearest** cluster,

$$z_i \leftarrow \text{argmin}_{z \in \{1 \ldots Z\}} \left\| \mathbf{x}_i - \boldsymbol{\mu}_z \right\|^2$$

2. Then, we put the cluster centres at the **mean** of the assigned datapoints,

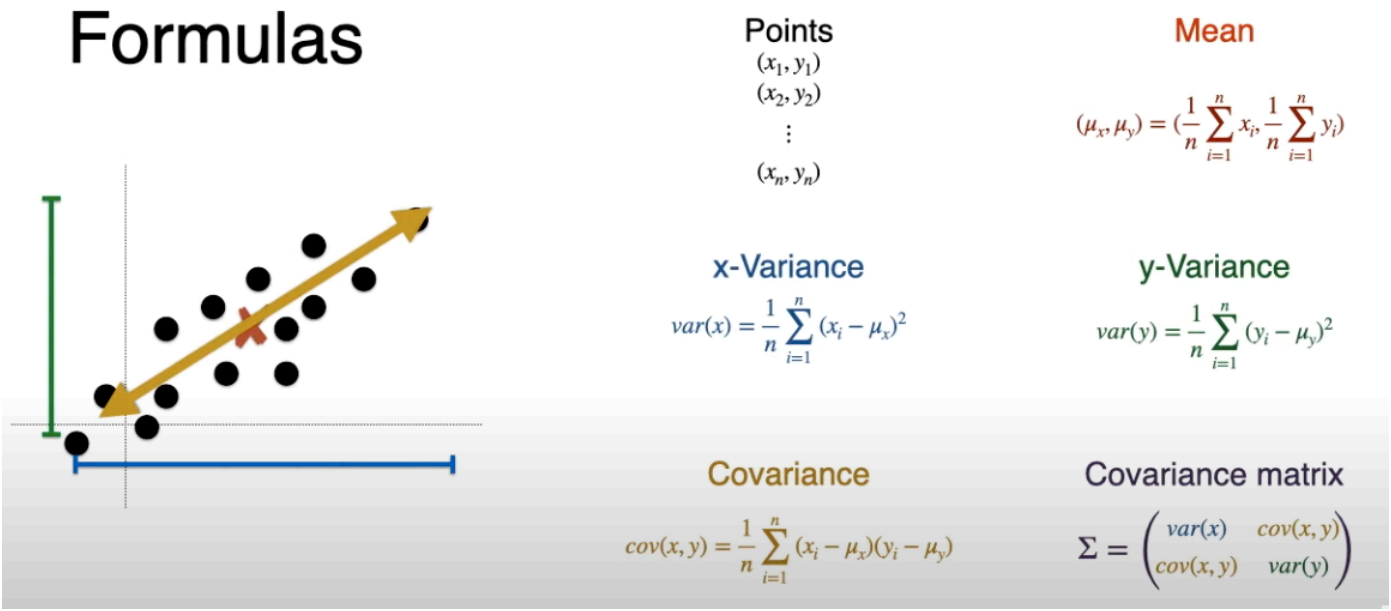
$$\boldsymbol{\mu}_z \leftarrow \frac{1}{\sum_i \delta_{z,z_i}} \sum_i \delta_{z,z_i} \mathbf{x}_i$$

$$\sum_i \delta_{z,z_i} = \text{number of datapoints in cluster } z$$

$$\sum_i \mathbf{x}_i \delta_{z,z_i} = \text{sum of all the datapoints in cluster } z$$

Covariance and correlation are very similar:

Better to use **n - 1** for **variance**



4. GMM (Gaussian Mixture Model) Soft Clustering

The goal of the GMM, is to model the density of the data we use the latent variables to give us a better model of the data.

$$\mathrm{P}(\mathbf{x}_i) = \sum_{z=1}^{Z} p_z \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z).$$

a density, with K different **Gaussians**, different locations, μk, and with different covariance matrices, Σk (sigma)

write this distribution in terms of a latent variable,

$$\mathrm{P}(\mathbf{x}_i) = \sum_{z_i=1}^{Z} \mathrm{P}(\mathbf{x}_i|z_i)\mathrm{P}(z_i)$$

where, P(zi) is a Categorical (i.e. a distribution over integers from 1 to Z), and represents the mixture component for the ith data point, and P(xi|zi) is a *sin

$$\mathrm{P}(z_i) = \mathrm{Categorical}(z_i; \mathbf{p}) = p_{z_i}$$
$$\mathrm{P}(\mathbf{x}_i|z_i) = \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_{z_i}, \boldsymbol{\Sigma}_{z_i}).$$

The substituting in the values of P(zi) and P(xi|zi), we get the same expression as above,

$$\mathrm{P}(\mathbf{x}_i) = \sum_{z_i=1}^{Z} \mathrm{P}(\mathbf{x}_i|z_i)\mathrm{P}(z_i) = \sum_{z_i=1}^{Z} p_{z_i}\mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_{z_i}, \boldsymbol{\Sigma}_{z_i}).$$

Now, we use Bayes theorem to give a **posterior distribution** over Zi, **(for soft clustering)** ⬇️⬇️⬇️⬇️

$$\mathrm{P}(z_i|\mathbf{x}_i) \propto \mathrm{P}(\mathbf{x}_i|z_i)\mathrm{P}(z_i)$$

5. EM Algorithm(similar to K-means)

- E step: updates the posterior over z_i for a given x_i and "saves" the posterior into q. The denominator normalizes the distribution so that it sums to 1.

$$q_{i;z} \leftarrow \mathrm{P}(z_i = z|x_i) = \frac{\mathrm{P}(\mathbf{x}_i|z_i = z)\mathrm{P}(z_i = z)}{\sum_{z'} \mathrm{P}(\mathbf{x}_i|z_i = z')\mathrm{P}(z_i = z')}$$

- M step: is a **weighted average** of x_i 's, with the weights corresponding to the degree to which that datapoint is assigned to that cluster.

$$\boldsymbol{\mu}_z \leftarrow \frac{\sum_i \mathbf{x}_i q_{i,z}}{\sum_i q_{i,z}}$$

Week 19 Feature Selection

0. Superposition

For a linear system: output of the linear combination of many input signals is the same linear combination of the outputs

1. revision

sine function: f(t) = asin2πμt

Frequency μ = 1/T

Amplitude A

Angular frequency ω = 2pi/T = 2pif

2. Trigonometric Fourier Seires (Frequency analysis)

Any **periodic** function can be expressed as :

$$f(x) = \sum_{n=0}^{\infty} a_n \cos\left(\frac{2\pi nx}{T}\right) + b_n \sin\left(\frac{2\pi nx}{T}\right)$$

N is the no. of cycles/period

a~n ~ and b_n are the Fourier Coefficients.

- The sinusoids are harmonically related: each one's frequency is **an integer** multiple of the fundamental frequency of the input signal.
- a_0 is often referred to as the DC term or the average of the signal.

$$f(x) = a_0 + \sum_{n=1}^{\infty} a_n \cos\left(\frac{2\pi nx}{T}\right) + b_n \sin\left(\frac{2\pi nx}{T}\right)$$

- The coefficients are:

$$a_n = \frac{2}{T} \int_{-T/2}^{+T/2} f(x) \cos\left(\frac{2\pi n x}{T}\right) dx$$

$$b_n = \frac{2}{T} \int_{-T/2}^{+T/2} f(x) \sin\left(\frac{2\pi n x}{T}\right) dx$$

3. 1D Fourier Transform

Fourier transform: Function that are **not periodic** can also be expressed as the integral of sines and/or cosines weighted by a coefficient.

$$F(u) = \int_{-\infty}^{\infty} f(x) \quad e^{-j2\pi u x} dx$$

Conversely, given F(u), we can obtain f(x):

$$f(x) = \int_{-\infty}^{\infty} F(u) \quad e^{j2\pi ux} du$$

Fourier Transform Pair a lossless representation of data

1D Fourier Transform: discrete form:

The Fourier Transform of a discrete function of one variable, *f(x), x=0,1,2...,N-1* is:

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x)\, e^{\frac{-j2\pi ux}{N}} \qquad \text{for } u = 0,1,2,\ldots,N-1.$$

Conversely, given $F(u)$, we can obtain $f(x)$ by means of the *inverse* Fourier Transform:

$$f(x) = \sum_{u=0}^{N-1} F(u)\, e^{\frac{j2\pi ux}{N}} \qquad \text{for } x = 0,1,2,\ldots,N-1.$$

These two equations are also known as the Fourier Transform Pair. Note, they constitute a lossless representation of data.

Euler's formula

$$e^{-j\theta} = \cos\theta - j\sin\theta$$

$$e^{ix} = \cos x + i \sin x$$

Then,

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) \left[\cos\left(\frac{2\pi ux}{N}\right) - j \sin\left(\frac{2\pi ux}{N}\right) \right]$$

$$\text{for } u = 0,1,2,\ldots,N-1.$$

We have transformed from a **time domain** to a **frequency domain** representation.

F(u) is a complex number & has real an imaginary parts:

$$F(u) = R(u) + jI(u)$$

Magnitude or spectrum of the FT:

$$|F(u)| = \sqrt{R^2(u) + I^2(u)}$$

Phase angle or phase spectrum:

$$\varphi(u) = \tan^{-1} \frac{I(u)}{R(u)}$$

Expressing F(u) in polar coordinates:

$$F(u) = |F(u)| e^{j\varphi(u)}$$

Week 20

1. 2D Fourier transform: continuous form

The Fourier Transform of a continuous function of two variables $f(x,y)$ is:

$$F(u,v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y) \quad e^{-j2\pi(ux+vy)} \; dxdy$$

Conversely, given $F(u,v)$, we can obtain $f(x,y)$ by means of the *inverse* Fourier Transform:

$$f(x,y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u,v) \quad e^{j2\pi(ux+vy)} \; dudv$$

These two equations are also known as the Fourier Transform Pair. Note, they constitute a lossless representation of data.

2. 2D Fourier transform: Discrete form

The FT of a discrete function of two variables, $f(x,y), x,y = 0,1,2 \ldots, N-1$, is:

$$F(u,v) = \frac{1}{N^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y)\ e^{-j2\pi(\frac{ux+vy}{N})} \quad \text{for } u,v = 0,1,2,\ldots,N-1\,.$$

Conversely, given $F(u,v)$, we can obtain $f(x,y)$ by means of the *inverse FT:*

$$f(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u,v)\ e^{j2\pi(\frac{ux+vy}{N})} \quad \text{for } x,y = 0,1,2,\ldots,N-1.$$

These two equations are also known as the Fourier Transform Pair.
Note, they constitute a lossless representation of data.

3. With Euler's Formula

$$F(u,v) = \frac{1}{N^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \left[\cos\left(\frac{2\pi(ux+vy)}{N}\right) - j\sin\left(\frac{2\pi(ux+vy)}{N}\right)\right]$$
$$\text{for } u,v = 0,1,2,\ldots,N-1\,.$$

We have transformed from a **time domain** to a **frequency domain** representation.

4. When u=0, v=0 the slowest varying frequency component

U = 0, v = 0 => average image graylevel

5.

$F(u,v)$ is a complex number & has real and imaginary parts:

$$F(u,v) = R(u,v) + jI(u,v)$$

Magnitude or *spectrum* of the FT:

$$|F(u,v)| = \sqrt{R^2(u,v) + I^2(u,v)}$$

Phase angle or phase spectrum:

$$\varphi(u,v) = \tan^{-1}\frac{I(u,v)}{R(u,v)}$$

Expressing $F(u,v)$ in polar coordinates:

$$F(u,v) = |F(u,v)|e^{j\varphi(u,v)}$$

Important property of the FT

6. Conjugate Symmetry

The FT of a real function f(x,y) gives:

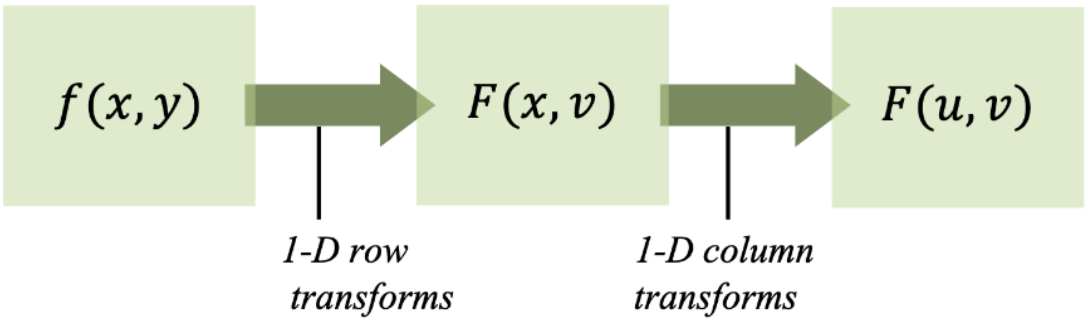
$$F(u,v) = F^*(-u,-v) \quad \Rightarrow \quad |F(u,v)| = |F^*(-u,-v)|$$

7. Separability

If a 2D transform is separable, the result can be found by successive application of two 1D transforms. This is a principle aspect of the **Fast Fourier Transform**(FFT).

Faster Computation: multiplying an $N\ x\ N$ image with an $m\ x\ m$ matrix would require N^2m^2operations. In 1D separable form, only $\Rightarrow N^2m$

$$F(u,v) = \frac{1}{N}\sum_{x=0}^{N-1} F(x,v) \quad e^{\frac{-j2\pi ux}{N}} \text{ where } F(x,v) = \frac{1}{N}\sum_{y=0}^{N-1} f(x,y) \quad e^{\frac{-j2\pi vy}{N}}$$



8. Rotation

Rotate the image and the Fourier space rotates.

$$f(r, \theta + \rho_0) \quad \Longrightarrow \quad F(\omega, \varphi + \rho_0)$$

9. Importance of Phase and Manipulating the Fourier Frequencies

---

Week 21

1. Feature Selection

Selecting a subset of the existing features without a transformation.

2. Feature Extraction

Transforming the existing features into a lower dimensional space.

3. Heuristic Feature Selection Methods

- **Bottom-up**: build up d features incrementally, starting with an empty set -> step-wise feature selection:
    - The best single feature is picked first
    - Then next best feature conditioned to the first

- **Top-down**: start with full set of feature and remove redundant ones successively -> step-wise feature elimination

4. Dimensionality Reduction

Sum of the variances = **sum of all eigenvalues** = 100% of variance in original data

The proportion of the variance that each eigenvector represents can be calculated by dividing the eigenvalue corresponding to that eigenvector by the sum of all eigenvalues.

Then the first d eigenvalues can be said to account for a fraction of the total variance in the data.

$$\frac{\sum_{i=1}^{d} \lambda_i}{\sum_{i=1}^{N} \lambda_i}$$

Convolution

1. Spacial Filtering

Many spatial filters are implemented with **convolution masks**.

Convolution mask is applied to each signal sample and its neighbourhood.

Convolution boosts or suppresses parts of a signal.

- f is the signal, h is the convolution filter
  - h has an origin
  - Normalization factor, e.g. $\frac{1}{5}$, is also part of the filter!

$\frac{1}{5}$ | -1 | 3 | -1 | Example 1D kernel

Normalisation factor: Take the **Absolute** of each value!!

2. 2D Convolution

- The discrete version of 2D convolution is defined as

$$g(x,y) = \sum_{m=-1}^{1} \sum_{n=-1}^{1} f(x-m, y-n)h(m,n)$$

Shorthand form:

$$g = f * h$$

3. 2D Correlation

- The discrete version of 2D correlation is defined as

$$g(x,y) = \sum_{m=-1}^{1} \sum_{n=-1}^{1} f(x+m, y+n)h(m,n)$$

Correlation = Convolution when **kernel is symmetric under 180 rotation**

4. Convolution Theorem:

Convolution in the **spatial** domain is equivalent to **multiplication** in the **frequency** domain and vice versa.

$$g(x,y) = f(x,y) * h(x,y) \quad \Longleftrightarrow \quad G(u,v) = F(u,v)\,H(u,v)$$

$$g(x,y) = f(x,y)\,h(x,y) \quad \Longleftrightarrow \quad G(u,v) = F(u,v) * H(u,v)$$