AINT351: Machine Learning

Lecture 4

Data modelling

2D Gaussian distribution

The joint Gaussian distribution for the vector y with mean μ and covariance matrix Σ is given by

$$p\left(\bar{y} \mid \bar{\mu},\Sigma\right)=\left|2\pi\Sigma\right|^{-\frac{1}{2}}\exp\left\{-\frac{1}{2}\left(\bar{y}-\bar{\mu}\right)^{T}\Sigma^{-1}\left(\bar{y}-\bar{\mu}\right)\right\}$$

Sometime written as

$$p\left(\bar{y} \mid \bar{\mu},\Sigma\right)=\mathrm{N}\left(\bar{\mu},\Sigma\right)$$

This equation says a lot!
Expanding the matrices and vectors leads to

$$p\left(\bar{y} \mid \mu,\Sigma\right)=\left|2\pi\begin{pmatrix}\Sigma_{11} & \Sigma_{12}\\ \Sigma_{21} & \Sigma_{22}\end{pmatrix}\right|^{-\frac{1}{2}}\exp\left\{-\frac{1}{2}\left(\begin{bmatrix}y_1\\ y_2\end{bmatrix}-\begin{bmatrix}\mu_1\\ \mu_2\end{bmatrix}\right)^{T}\begin{pmatrix}\Sigma_{11} & \Sigma_{12}\\ \Sigma_{21} & \Sigma_{22}\end{pmatrix}^{-1}\left(\begin{bmatrix}y_1\\ y_2\end{bmatrix}-\begin{bmatrix}\mu_1\\ \mu_2\end{bmatrix}\right)\right\}$$

2D Gaussian distribution

The covariance determinant and inverse terms can be written as

$$\left|\begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}\right| = \Sigma_{11}\Sigma_{22} - \Sigma_{12}\Sigma_{21}$$

$$\begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}^{-1} = \frac{1}{\Sigma_{11}\Sigma_{22} - \Sigma_{12}\Sigma_{21}} \begin{pmatrix} \Sigma_{22} & -\Sigma_{12} \\ -\Sigma_{21} & \Sigma_{11} \end{pmatrix}$$

Therefore

$$p(\bar{y} \mid \bar{\mu}, \Sigma) = \left(2\pi\left(\Sigma_{11}\Sigma_{22} - \Sigma_{12}\Sigma_{21}\right)\right)^{-\frac{1}{2}} \exp\left\{ -\frac{1}{2} \begin{bmatrix} y_1 - \mu_1 \\ y_2 - \mu_2 \end{bmatrix}^T \begin{pmatrix} \frac{\Sigma_{22}}{\left(\Sigma_{11}\Sigma_{22} - \Sigma_{12}\Sigma_{21}\right)} & \frac{-\Sigma_{12}}{\left(\Sigma_{11}\Sigma_{22} - \Sigma_{12}\Sigma_{21}\right)} \\ \frac{-\Sigma_{21}}{\left(\Sigma_{11}\Sigma_{22} - \Sigma_{12}\Sigma_{21}\right)} & \frac{\Sigma_{11}}{\left(\Sigma_{11}\Sigma_{22} - \Sigma_{12}\Sigma_{21}\right)} \end{pmatrix} \begin{bmatrix} y_1 - \mu_1 \\ y_2 - \mu_2 \end{bmatrix} \right\}$$

2D independent Gaussian distribution

If the two components of the vector y are independent then

$$\Sigma_{12} = 0$$

$$\Sigma_{21} = 0$$

Therefore

$$p(\bar{y} \mid \bar{\mu}, \Sigma) = \left(2\pi\Sigma_{11}\Sigma_{22}\right)^{-\frac{1}{2}} \exp\left\{ -\frac{1}{2} \begin{bmatrix} y_1 - \mu_1 \\ y_2 - \mu_2 \end{bmatrix}^T \begin{pmatrix} \frac{1}{\Sigma_{11}} & 0 \\ 0 & \frac{1}{\Sigma_{22}} \end{pmatrix} \begin{bmatrix} y_1 - \mu_1 \\ y_2 - \mu_2 \end{bmatrix} \right\}$$

Giving

$$p(\bar{y} \mid \bar{\mu}, \Sigma) = \left(2\pi\Sigma_{11}\Sigma_{22}\right)^{-\frac{1}{2}} \exp\left(-\frac{1}{2} \left(\frac{\left(y_1 - \mu_1\right)^2}{\Sigma_{11}} + \frac{\left(y_2 - \mu_2\right)^2}{\Sigma_{22}} \right) \right)$$

Which is the product of two 1D Gaussians

$$p(\bar{y} \mid \bar{\mu}, \Sigma) = \frac{1}{\sqrt{2\pi\Sigma_{11}}} \exp\left(-\frac{1}{2} \frac{\left(y_1 - \mu_1\right)^2}{\Sigma_{11}} \right) \frac{1}{\sqrt{2\pi\Sigma_{22}}} \exp\left(-\frac{1}{2} \frac{\left(y_2 - \mu_2\right)^2}{\Sigma_{22}} \right)$$

1D Gaussian data with arbitrary μ and σ

To generate 1 iD Gaussian with $\mu = 0$ and $\sigma=1$ we can use the Matlab randn function:

```
data = randn(1,1);
```

To change to sample drawn from a distribution with non-zero mean and non-unity standard deviation we need to scale by σ and hen add on μ

```
dataNew = data * σ + μ;
```

Note in 1D case

$\sigma = \sqrt{(var)}$;

where var is the variance.
In the multidimensional case we have a covariance matrix not a scalar value

Generate a ND Gaussian distribution

Question: If we use randn(N,1) to draw N samples from a 1D distribution x_1 and use if again to draw another N samples from a 1D distribution x_2 and then build a 2D vector X, what is to covariance matrix of the 2D dataset X?

$$\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

What are the covariance terms equal to here?

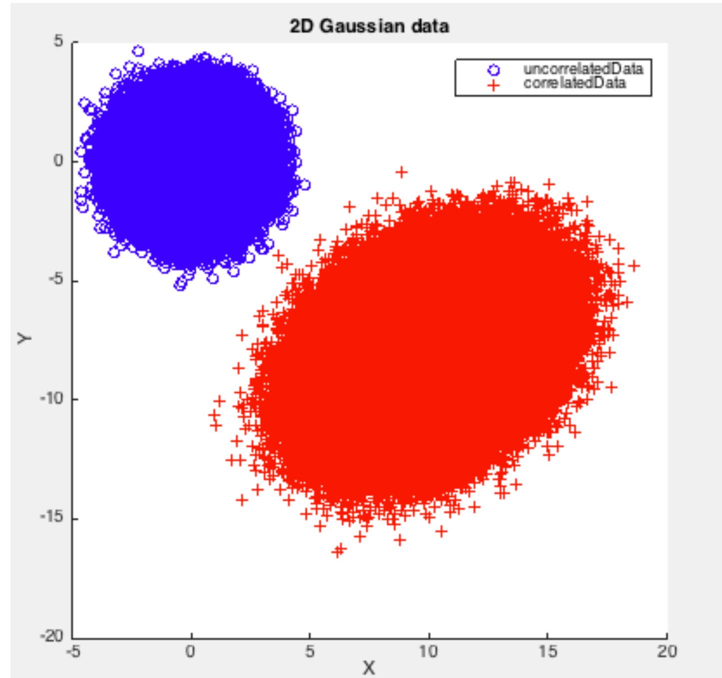
$$\Sigma_{12} = 0$$
$$\Sigma_{21} = 0$$

So how can we generate a joint Gaussian with covariance matrix 'K' and mean vector 'meanVal' ?

Generate correlated ND Gaussian data

To generate a random sample from a 2-dimensional joint Gaussian with covariance matrix 'K' and mean vector 'meanVal'

- First need to decompose the covariance matrix such that $A^T A = K$
- We can use Cholesky decomposition to do this
- Then multiply by A and add on meanVal:




```
% select large number of samples
samples = 1000000;

% example mean and covariance
meanVal = [10 -8]';
K       = [3 1; 1 3;];

% generate 2-D uncorrelated data of length 'samples'
uncorrelatedData = randn(2,samples);

% generate correlated data with covariance 'K' and specified mean 'meanVal'
correlatedData= chol(K) * uncorrelatedData + repmat(meanVal,1,samples);
```

AINT351: Machine Learning

Lecture 4

Learning from data

Terminology for types of learning

- Maximum likelihood (MP) learning

Does not assume a prior over the model parameters. Finds q parameter settings that maximizes the likelihood of the data $P(D|\theta)$

- Maximum a posteriori (MAP) learning

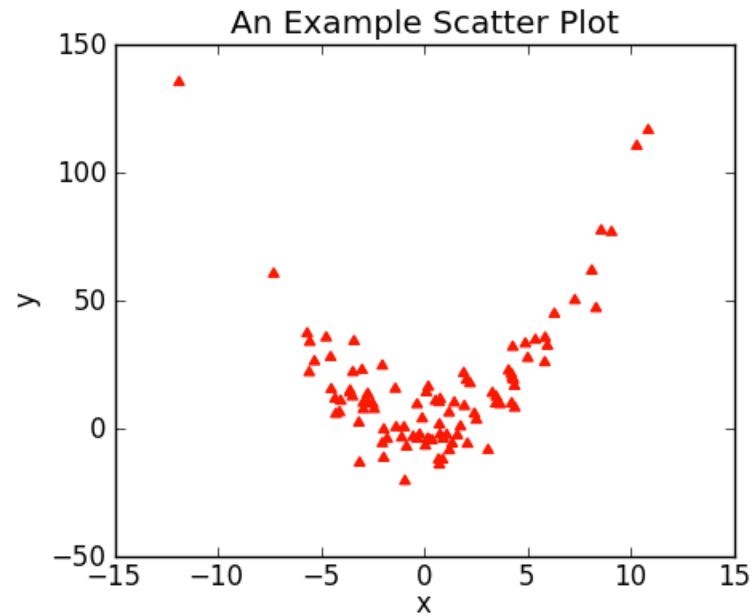
Assumes a prior over the model parameters $P(\theta)$. Finds a parameter settings that maximizes the posterior $P(\theta|D)\ \alpha\ P(\theta)\ P(D|\theta)$

- Bayesian learning

Assumes a prior over the model parameters . Computes the posterior of the parameters $P(\theta|D)$

Simple statistical modeling

- Assume we have a dataset $Y = \{y_1, \ldots, y_N\}$
- Each data point is a vector of D features
  $y_i = \{y_{i1}, \ldots, y_{iD}\}$
- The data points are I.I.D (independent and identically distributed)
- One of the simplest forms of unsupervised learning is to model the mean and correlations between the D features of the data.
- We can do so using the multivariate Gaussian model



$$p(\bar{y} \mid \bar{\mu}, \Sigma) = |2\pi\Sigma|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(\bar{y}-\bar{\mu})^T \Sigma^{-1} (\bar{y}-\bar{\mu})\right\}$$

Joint probability of a dataset

If two events y_1 and y_2 are independent we know that their joint probability is given by:

$$p(y_1, y_2) = p(y_1)\,p(y_2) = \prod_{n=1}^{2} p(y_n)$$

Similarly if data points in the dataset Y= $\{y_{1,}$ .... $y_N\}$ are I.I.D (independent and identically distributed) then, the likelihood of this dataset is

$$p(Y) = \prod_{n=1}^{N} p(y_n)$$

ML estimation of a Gaussian

Given the dataset Y= {$y_{1, \ldots} \; y_N$}, the likelihood of this dataset is

$$p(Y \mid \mu, \Sigma) = \prod_{n=1}^{N} p(y_n \mid \mu, \Sigma)$$

We wish to find the maximum likelihood of the dataset
⇔ maximize log likelihood (because mathematically its easier)

$$L = \log \prod_{n=1}^{N} p(y_n \mid \mu, \Sigma) = \sum_{n=1}^{N} \log\left(p(y_n \mid \mu, \Sigma)\right)$$

ML estimation of a Gaussian

Since

$$p(y \mid \mu, \Sigma) = |2\pi\Sigma|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(y-\mu)^T \Sigma^{-1} (y-\mu)\right\}$$

Substituting into the expression for likelihood

$$L = \sum_{n=1}^{N} \log\left(p(y_n \mid \mu, \Sigma)\right)$$

Therefore

$$L = -\frac{N}{2}\log|2\pi\Sigma| - \frac{1}{2}\sum_{N}(y_n-\mu)^T \Sigma^{-1} (y_n-\mu)$$

Minimize -L

We wish to find the maximum likelihood, so minimize –L

$$-L = \frac{N}{2}\log\left|2\pi\Sigma\right| + \frac{1}{2}\sum_{N}\left(y_n-\mu\right)^T\Sigma^{-1}\left(y_n-\mu\right)$$

We differentiate -L w.r.t to the parameters, leading to

$$\frac{\partial L}{\partial \mu} = 0 \Rightarrow \tilde{\mu} = \frac{1}{N}\sum_{N} y_n$$

sample mean

$$\frac{\partial L}{\partial \Sigma} = 0 \Rightarrow \tilde{\Sigma} = \frac{1}{N}\sum_{N}\left(y_n-\mu\right)^T\left(y_n-\mu\right)$$

sample covariance

Calculate mean and covariance in Matlab

$$\tilde{\mu} = \frac{1}{N}\sum_{N} y_n$$

sample mean

```
% compute sum and divide by length
mm = sum(AS,2)/length(AS)

% use mean command
m = mean(AS,2)
```

$$\tilde{\Sigma} = \frac{1}{N}\sum_{N}\left(y_n - \mu\right)^T\left(y_n - \mu\right)$$

sample covariance

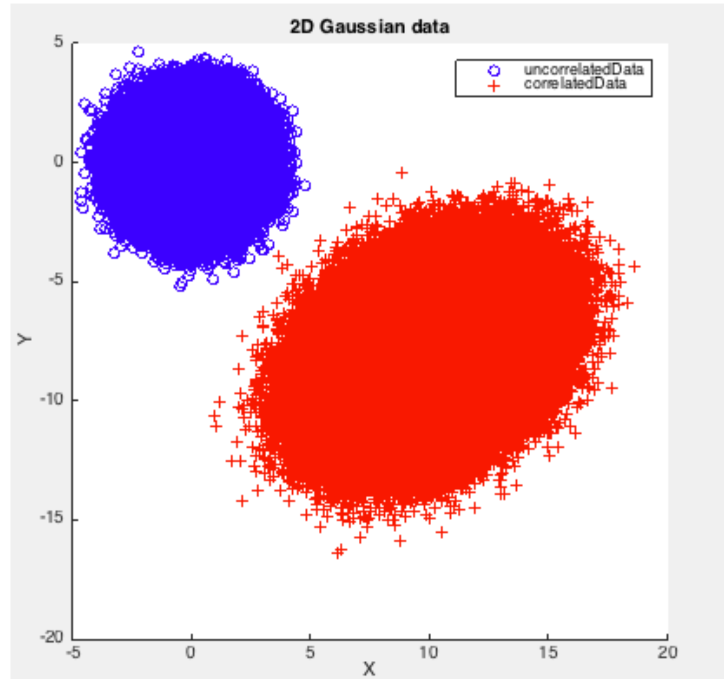
```
% subtract mean from data points and then multiply by a transposed copy
cc = (correlatedData - repmat(mm,1,samples))*(correlatedData - repmat(mm,1,samples))'/length(correlatedData)

% use cov command
c = cov(correlatedData')
```

Why do we care about this?

- Can now fit a single Gaussian to our dataset!

```
% use mean command
m = mean(correlatedData,2)

% use cov command
c = cov(correlatedData')
```

```
m =

   10.0014
   -7.9995


c =

    3.3294    0.9368
    0.9368    2.6680
```



Later we will fit more sophisticated models, so understanding the simplest is very helpful!

Gaussian class-conditional model

$$p(\bar{y} \mid \bar{\mu}, \Sigma) = |2\pi\Sigma|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(\bar{y}-\bar{\mu})^T \Sigma^{-1} (\bar{y}-\bar{\mu})\right\}$$

- The maximum likelihood fit of a Gaussian to some data is the Gaussian whose mean is equal to the data mean and whose covariance is equal to the sample covariance.

- One very nice feature of this model is that the maximum likelihood parameters can be found in closed-form, so we don't have to use iterative solutions

- Seems easy.
- And works surprisingly well.
- But we can do even better with some simple regularization

Three limitations

- We cannot account for higher order statistical structure in the data
  - These require nonlinear and hierarchical models

- We need to deal with outliers
  - These require nonlinear and hierarchical models

- The multivariate model uses D(D+1)/2 parameters.
  - If D is very large we need to use dimensionality reduction

AINT351: Machine Learning

Lecture 4

Bayesian learning

Frequentist and Bayesian statistics

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

- Bayesian approach
  - Probability is a “degree of confidence” that one attaches to an uncertain event
  - Requires a priori estimation of the model's likelihood, naturally incorporating prior knowledge

ML learning

$$h_{ML} = \underset{h \in H}{argmax}\ P(D|h)$$

- P (D|h) is often called the likelihood of D given h
- Here it is assumed that every hypothesis is equally probable a priori

ML estimation of coin flip probability

- We have a coin and wish to estimate the outcome (head or tail) from observing a series of coin tosses. *Let:*

- θ = probability of tossing a head
- Therefore probability of tail= (1- θ)
- Let h be the number of heads
- Let n be the total number of trials.
- Assume I.I.D (coin doesn't change between flips)
- The likelihood of throwing h heads, independent or their order is given by:

$$L(\theta) = \theta^h (1-\theta)^{n-h}$$

$$\Rightarrow \log L(\theta) = h \log \theta + (n-h) \log(1-\theta)$$

ML estimation of coin flip probability

To find the ML estimate for θ we look at when $dL/d\theta = 0$:

$$\frac{d}{d\theta}\log L(\theta) = \frac{h}{\theta} - \frac{n-h}{1-\theta} = 0 \quad \Rightarrow \frac{h}{\theta} = \frac{n-h}{1-\theta} \quad \Rightarrow h(1-\theta) = (n-h)\theta$$

$$\Rightarrow \theta_{ML} = \frac{h}{n}$$

Thus we should divide number of heads by total number of trials.

In a given experiment, the first flip may result in a tails

In this case ML estimate predicts zero probability of seeing heads!
But we know this cannot be the case!

Flip coin in a Matlab simulation

- Question: How can we simulate a coin flip in Matlab?

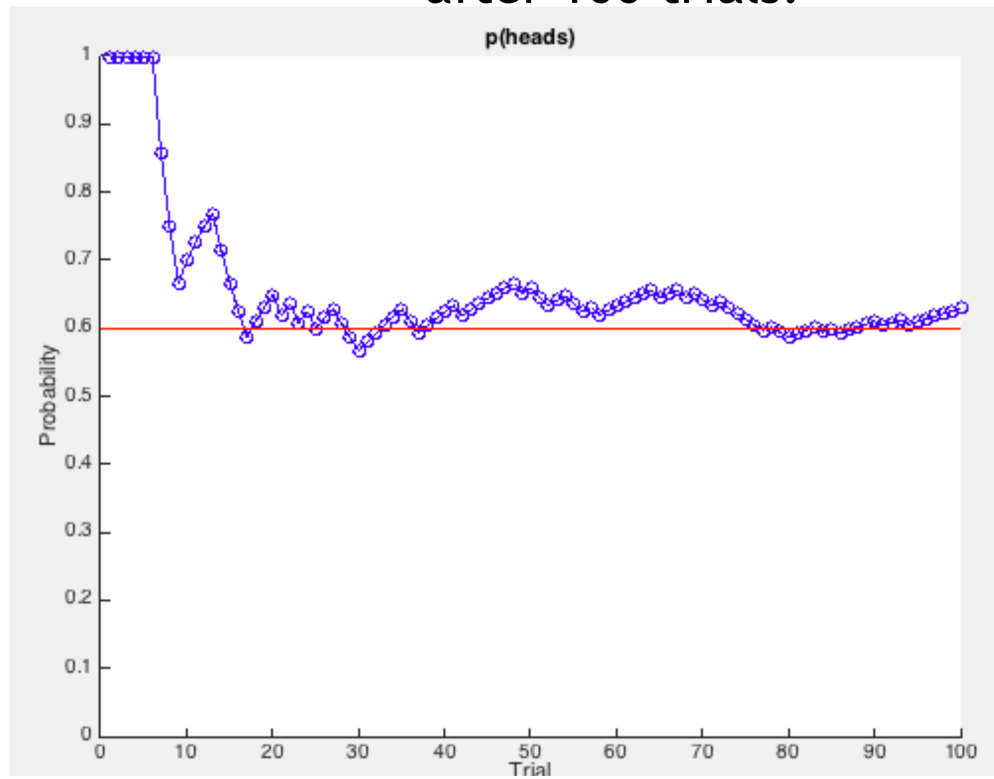
```
function wasHeads = FlipCoin(coinBias)
% flip coin and return heads/tails outcome

val = rand(1,1);
if(val < coinBias)
    wasHeads=1;
else
    wasHeads=0;
end
```

ML estimation of coin flip probability

Use a Matlab simulation with coin bias = 0.6

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% frequentist update
numberOfFlips = 100;
trials=0;
heads=0;
pHeads=[];
for idx=1:numberOfFlips
    % flip coind
    wasHeads = FlipCoin(coinBias);
    if(wasHeads)
        heads=heads+1;
    end
    trials=trials+1;
    % estimate estimate of heads
    pHeads(idx) = (heads)/ trials;
    flip(idx)=idx;
end
```

Might get something like this after 100 trials:



If get several heads in row then can get inappropriate bias to heads
Can we do better than this?
Here we only have point estimate

Remember Bayes theorem

- Generally we want to find the most probable hypothesis given the training data

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

- P (h) prior probability of h, represents our belief in what h should be
- P (D) prior probability of D
- P (D|h) probability of observing D given h holds
- P (h|D) posterior probability of h after D has been observed

This leads to MAP learning

- Really want the most probable hypothesis given the training data

$$
\begin{aligned}
h_{MAP} &= \underset{h \in H}{argmax}\ P(h|D) \\
&= \underset{h \in H}{argmax}\ \frac{P(D|h)P(h)}{P(D)} \\
&= \underset{h \in H}{argmax}\ P(D|h)P(h)
\end{aligned}
$$

- P (D) can be dropped, because it is a constant and independent of h
- NB: MAP and ML estimate are identical when the prior is uniformly distributed

MAP estimation of coin flip probability

- Rather than estimating a single θ, we obtain a distribution over possible values of θ
- Consider θ = probability of tossing a head as a random variable
- We want to take our prior belief of what θ should be into account
- Using Bayes theorem we can write the posterior is given by:

$$p(\theta = x \mid D) = \frac{p(D \mid \theta = x)\, p(\theta = x)}{p(D)}$$

Where

$p(D \mid \theta = x)$ is the same as expression from ML estimate with θ fixed to value x

And

$p(\theta = x)$ Is the probability θ around x without seeing the data – the prior

How should be choose the prior?

- Since prior is a distribution then:

$$\int_X p(\theta = x)dx = 1$$

$$or$$

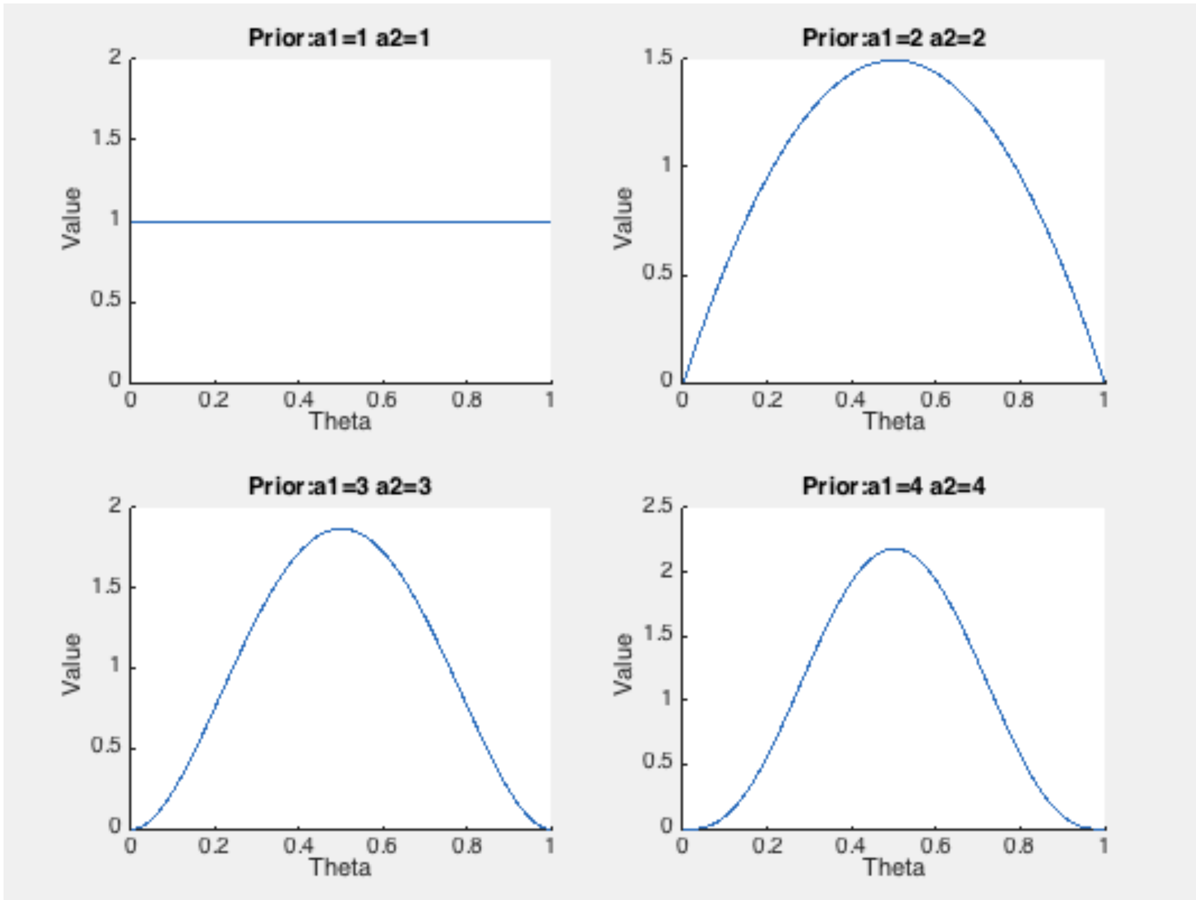
$$\int p(\theta)d\theta = 1$$

$$where$$

$$\theta \in [0,1]$$

How should be choose the prior?

- Want function that can represent the following kind of distributions



- A suitable function for the prior is the Beta distribution since
It captures some of these important properties

Acknowledgements

- Zoubin Ghahramani
- Patrick Lam
- Wikipedia
- Sam Roweis
- http://nucinkis-lab.cc.ic.ac.uk/HELM/helm_workbooks.html