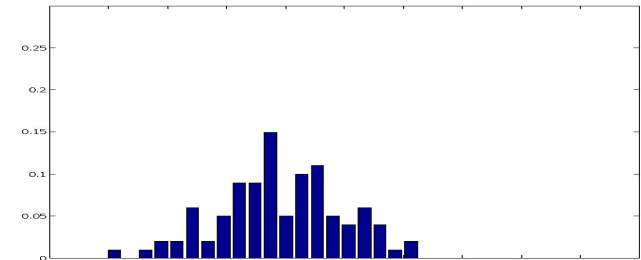
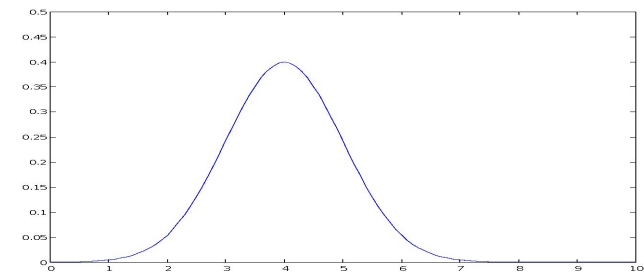


Density Estimation

Data histogram



Learned density



Model the density of the probability from which x is drawn. $\mathbf{x} \sim p(\mathbf{x}; \mathbf{w})$, where \mathbf{w} represents an unknown parameter vector that needs to be learned.

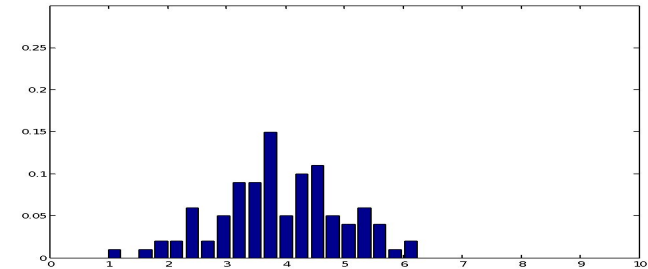
Decision theory, abnormal event detection.

Density Estimation

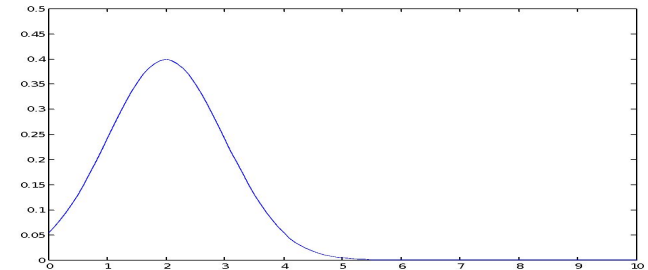
Which model is best?

Why?

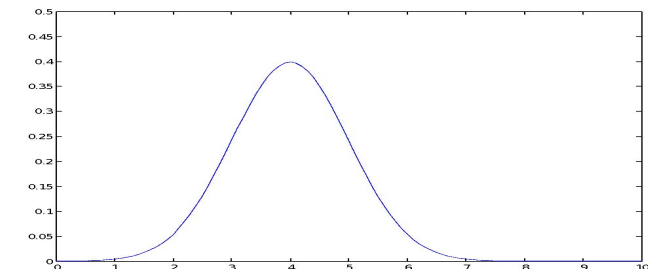
Data histogram



Model 1



Model 2



Density Estimation

How do we obtain the model parameters?

Which model is the best?

Suppose we have a *family* of probability models for $\mathbf{x} \sim p(\mathbf{x}; \mathbf{w})$, where \mathbf{w} represents an unknown parameter vector.

We will consider two forms of learning:

- o Maximum Likelihood
- o Bayesian learning

We will consider Gaussians models. Formally

Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ we wish to fit a density $p(\mathbf{x}) = N_{\mathbf{x}}(\boldsymbol{\mu}, \Sigma)$.

$$\mathbf{x}_n \in \mathbb{R}^d, \boldsymbol{\mu} \in \mathbb{R}^d, \Sigma \in \mathbb{R}^{d \times d}$$

Maximum Likelihood estimation

Assuming samples are independent...

$$p(\chi | \mathbf{w}) = \prod_{n=1}^N p(\mathbf{x}_n | \mathbf{w}) = L(\mathbf{w})$$

$L(\mathbf{w})$ is the likelihood of the observed data given the parameter \mathbf{w}

Intuitively choosing \mathbf{w} that makes $L(\mathbf{w})$ large is good (makes the data more likely). So ...

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}} L(\mathbf{w}) \quad (\text{maximum likelihood})$$

we could argue (and will) that $p(\mathbf{x} | \mathbf{w})$ is not the relevant quantity (we know the data exists with probability 1!)

Maximum Likelihood fitting

In practice we minimise:

$$E = -\ln L(\mathbf{w}) = -\sum_n \ln p(\mathbf{x}_n | \mathbf{w})$$

the *negative log likelihood*.

General ML estimates will require *numerical optimization*.

Depending on model structure we could use:

- Gradient descent (back prop.)
- Analytically solve for $\frac{\nabla L}{\nabla \mathbf{w}} = 0$

or alternatives (not covered in ELEM041):

- Levenberg Marquardt
- Conjugate gradient

ML fitting a single Gaussians

A simple example of *unsupervised learning* is fitting a Gaussian distribution to some data. In one dimension:

let $\mathcal{X} = \{x_1, x_2, \dots, x_N\}$ we wish to fit a density $p(x) = N_x(\mu, \sigma^2)$

$$E = -\ln L(\mathbf{w}) = \sum_n \left(\frac{(x_n - \mu)^2}{2\sigma^2} + \ln \sigma + \frac{1}{2} \ln 2\pi \right).$$

Quadratic in μ , hence ML estimate $\hat{\mu}$:

$$\left. \frac{\partial E}{\partial \mu} \right|_{\hat{\mu}} = -\sum_n \frac{(x_n - \hat{\mu})}{\sigma^2} = 0$$

$$\rightarrow \hat{\mu} = \frac{1}{N} \sum_n x_n \quad (\text{sample mean!})$$

ML fitting a single Gaussian

To solve for ML estimate of σ we use the fact that we already know $\hat{\mu}$:

$$\frac{\partial E}{\partial \sigma} = \sum_n \left(\frac{1}{\sigma} - \frac{(\mathbf{x}_n - \hat{\mu})^2}{\sigma^3} \right)$$

Equating to zero gives :

$$\sigma^2 = \frac{1}{N} \sum_n (\mathbf{x}_n - \hat{\mu})^2 \quad \text{(sample variance)}$$

Recall that the sample variance is a *biased* estimate for variance. However such ML estimates are consistent and asymptotically optimal.

ML fitting a single Gaussians

In higher dimensions $p(x | \mu, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$

we get the similar results of:

ML estimate for mean $\hat{\mu}$:

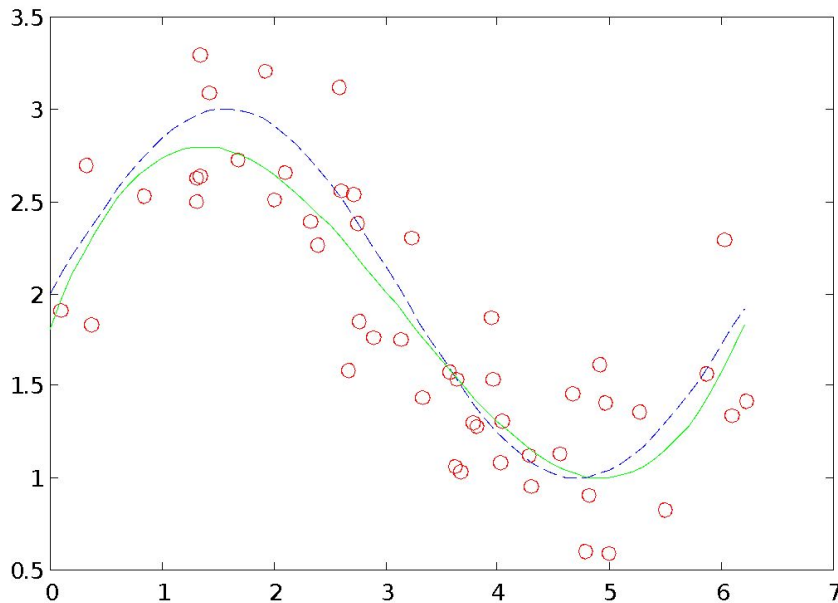
$$\hat{\mu} = \frac{1}{N} \sum_n \mathbf{x}_n \quad (\text{sample mean})$$

and ML estimate for covariance matrix:

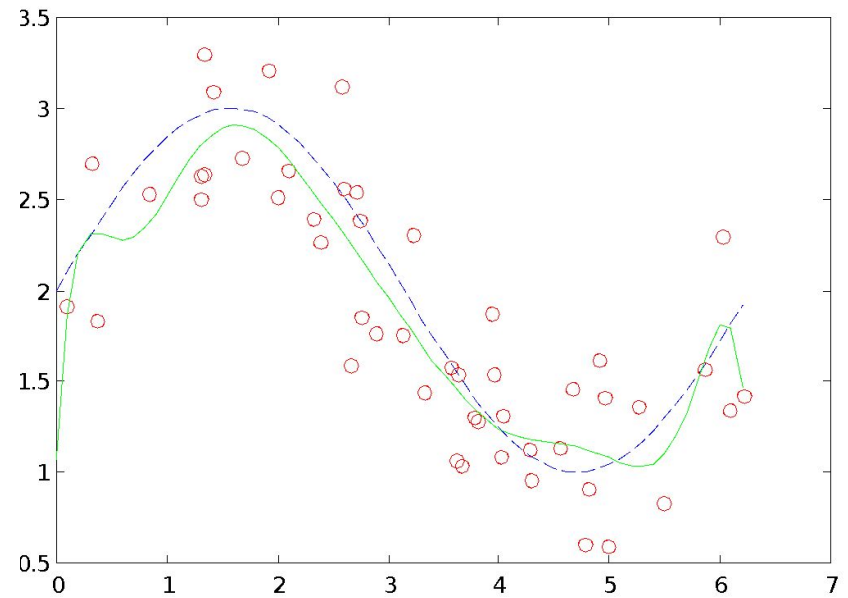
$$\hat{\Sigma} = \frac{1}{N} \sum_n (\mathbf{x}_n - \hat{\mu})(\mathbf{x}_n - \hat{\mu})^T \quad (\text{sample covariance})$$

Overfitting in ML (regression)

Given a limited data set maximising $L(\mathbf{w})$ may lead to *overfitting*. If the model order (dim. \mathbf{w}) is large enough we can 'exactly' fit model to data.



3rd order polynomial fit



10th order polynomial fit

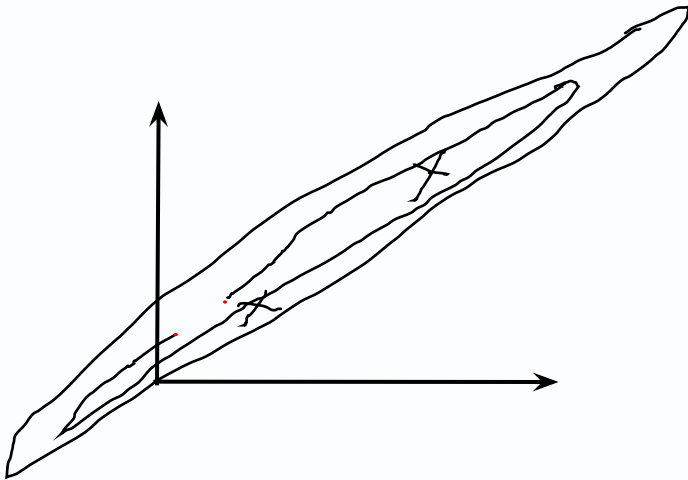
Overfitting in ML density estimation

Let $X = \{x_1, x_2, \dots, x_N\}$ we wish to fit a density $p(x) = N_x(\mu, \Sigma)$.

$$x_n \in \mathbb{R}^d, \mu \in \mathbb{R}^d, \Sigma \in \mathbb{R}^{d \times d}$$

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

In the case that the data dimensionality is higher than the data samples the covariance matrix is singular. Non invertible, zero determinant.



Overfitting in ML density estimation

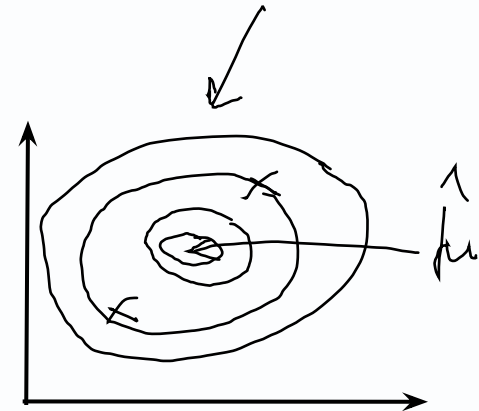
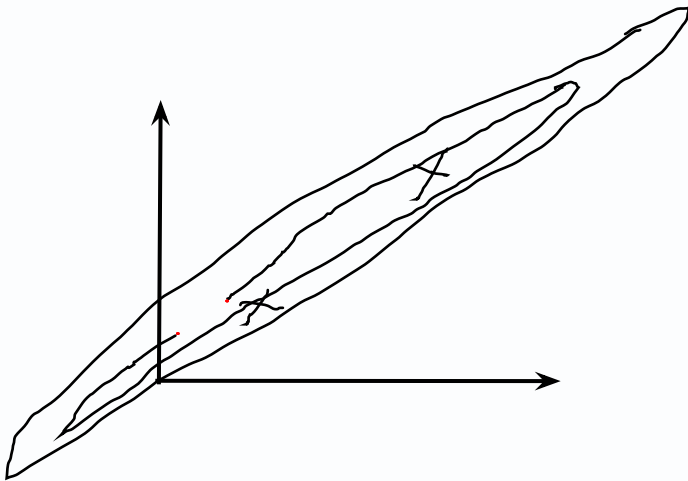
$$p(x | \mu, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

1. Put constraints in the form of the covariance matrix. E.g.

$$\Sigma = \sigma I = \begin{bmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & \sigma \end{bmatrix}$$

$$\sigma^2 = \sum_n \left(x_n - \bar{\mu} \right)^T \left(x_n - \bar{\mu} \right)$$

Concentric circles



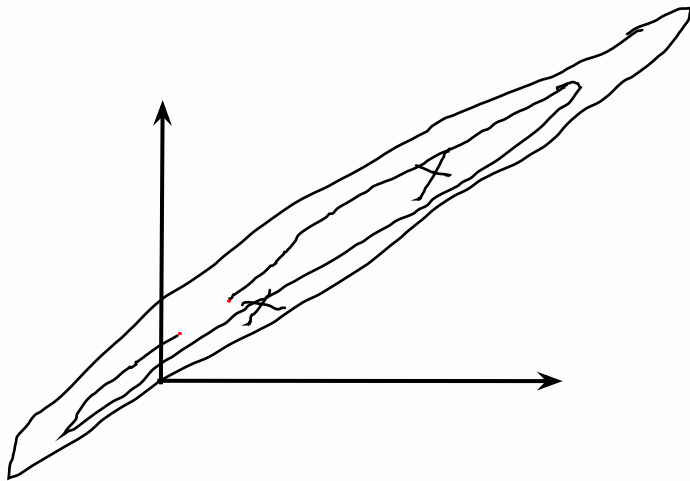
Overfitting in ML density estimation

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

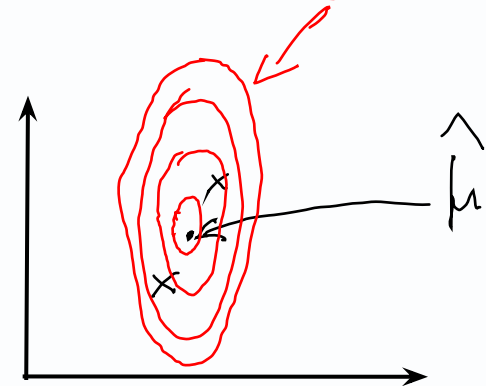
2. Put constraints in the form of the covariance matrix. E.g.

$$\Sigma = \begin{bmatrix} \sigma_1 & \dots & \sigma_d \end{bmatrix} I = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_d \end{bmatrix}$$

$$\sigma_j^2 = \sum_n \left(x_n(j) - \mu(j) \right)^2$$



Axes aligned ellipse



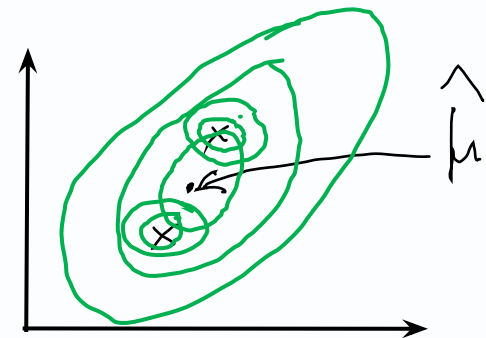
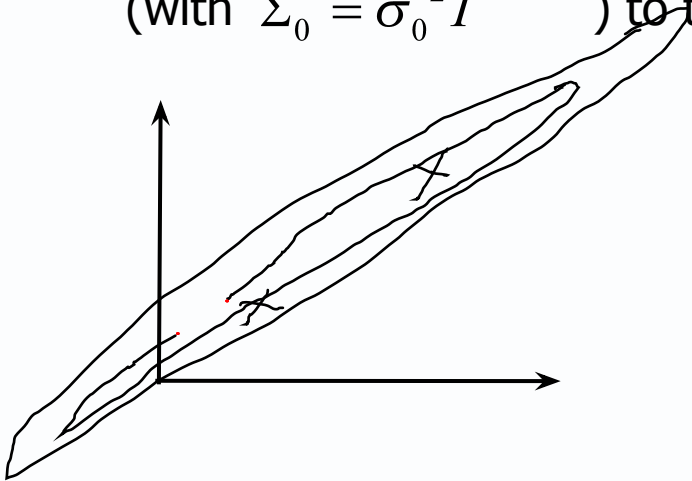
Overfitting in ML density estimation

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

2. Regularise the covariance matrix by adding a diagonal matrix $\Sigma_0 = \sigma_0^2 I$

$$\hat{\Sigma} = \sum_n (x_n - \hat{\mu})(x_n - \hat{\mu})^T + \sigma_0^2 I = \begin{bmatrix} \sum_n (x_n(1) - \hat{\mu}(1))^2 + \sigma_0^2 & \dots \\ \vdots & \ddots \\ \dots & \dots & \sum_n (x_n(d) - \hat{\mu}(d))^2 + \sigma_0^2 \end{bmatrix}$$

Leads to "thicker" Gaussian. Equivalent to adding zero mean Gaussian noise (with $\Sigma_0 = \sigma_0^2 I$) to the original dataset.



Bayesian learning I

ML generates an estimator \mathbf{w} for $p(\mathbf{x})$ from a family of densities $p(\mathbf{x} | \mathbf{w})$ by optimizing $L(\mathbf{w})$. However the data already exists!

We are really interested in the probability of \mathbf{w} given the data.
This extends the notion of RV to the parameter \mathbf{w} .

Bayes rule gives :

$$p(\mathbf{w} | \chi) = \frac{p(\chi | \mathbf{w}) p(\mathbf{w})}{p(\chi)}$$

Requires a prior distribution on \mathbf{w} , $p(\mathbf{w})$. Note also the role played by the likelihood, $p(\chi | \mathbf{w})$.

Bayesian learning II

We can also evaluate the probability of new data

$$p(x^{(\text{new})} | \chi) = \int p(x | \mathbf{w}) p(\mathbf{w} | \chi) d\mathbf{w}$$

Not just one value of \mathbf{w} is used, instead a weighted average of values (this is the basis of the popular Kalman filter).

Bayesian Inference \rightarrow Integration NOT optimization

In practice integrating and combining densities is difficult. One option is to use *conjugate priors* ($\mathbf{p}(\mathbf{w})$ is a conjugate prior if the posterior $\mathbf{p}(\mathbf{w} | \chi)$ has the same functional form – see the Gaussian example below).

Modern Bayesian methods have concentrated on using Monte Carlo integration (computationally intensive but very general).

Bayesian learning: 1-d Gaussians

Suppose $p(x | \mu) \sim N_x(\mu, \sigma^2)$ where σ^2 is known. We wish to estimate μ from data. Assume Gaussian prior on μ :

$$p(\mu) \sim N_\mu(\mu_0, \sigma_0^2)$$

(intuitively we believe that μ belongs within some range of values)

Inferring μ given some data : $\chi = \{x_1, x_2, \dots, x_N\}$:

$$p(\mu | \chi) = \frac{p(\chi | \mu) p(\mu)}{p(\chi)}$$

Product of Gaussians = Gaussian (why?)

$$-\ln p(\mu | \chi) = \underbrace{\left[\sum_n \frac{(x_n - \mu)^2}{2\sigma^2} \right]}_{-\text{loglikelihood}} + \underbrace{\left[\frac{(\mu - \mu_0)^2}{2\sigma_0^2} \right]}_{-\text{logprior}} + \text{const.}$$

Still quadratic in μ , hence Gaussian : $p(\mu | \chi) = N_\mu(\mu_N, \sigma_N^2)$

Bayesian learning: 1-d Gaussians

Equating the quadratic terms, μ^2 :

$$\frac{\mu^2}{2\sigma_N^2} = \frac{N\mu^2}{2\sigma^2} + \frac{\mu^2}{2\sigma_0^2}$$

giving

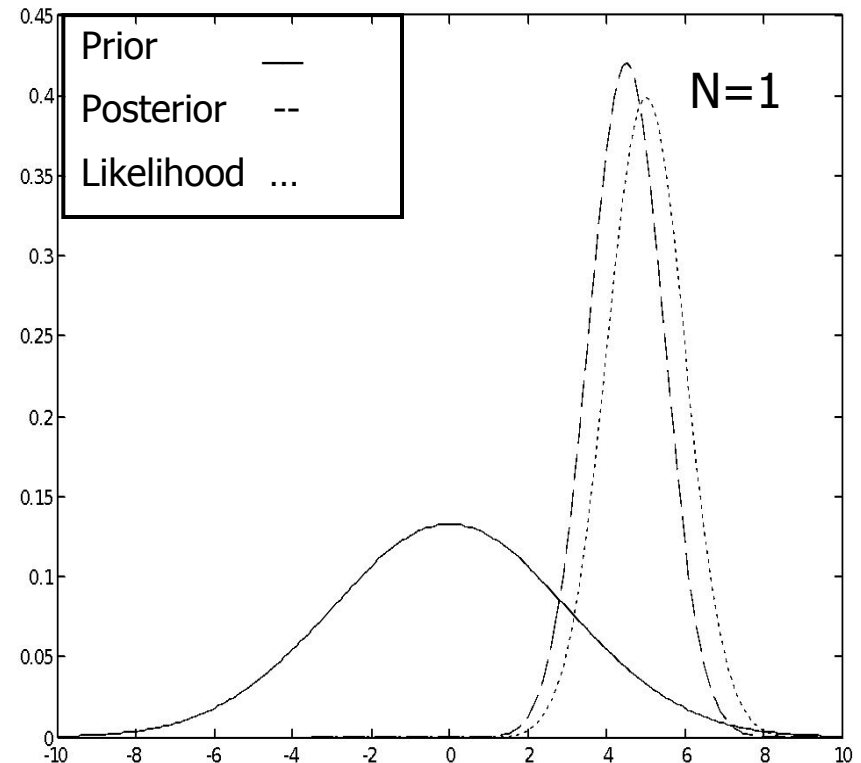
$$\sigma_N^2 = \left(\frac{N}{\sigma^2} + \frac{1}{\sigma_0^2} \right)^{-1}$$

and the linear terms, μ :

$$\frac{2\mu_N\mu}{2\sigma_N^2} = \frac{2\mu_0\mu}{2\sigma_0^2} + \frac{1}{2\sigma^2} \sum_n 2x_n\mu$$

giving :

$$\mu_N = \frac{N\sigma_0^2}{N\sigma_0^2 + \sigma^2} \bar{x} + \frac{\sigma^2\mu_0}{N\sigma_0^2 + \sigma^2}$$



Bayesian choice of \mathbf{w}

Often we may wish to get a single value for \mathbf{w} as an estimate from some data. From a Bayesian perspective this involves defining a loss function (c.f. Bayesian decision theory). 2 common choices are:

1. *Posterior Mean*:

$$\mathbf{w} = E\{\mathbf{w} \mid \chi\} = \int \mathbf{w} p(\mathbf{w} \mid \chi) d\mathbf{w}$$

this comes from a Mean Squared Error loss function.

2. *Maximum a Posteriori* (MAP) estimates:

$$\mathbf{w} = \operatorname{argmax}[p(\mathbf{w} \mid \chi)]$$

closely related to ML estimation – not really *very* Bayesian (comes from a 0-1 loss function)

The Bayes - ML link

We can see that ML and Bayesian learning are linked through the likelihood :

$$p(\mathbf{w} | \chi) \propto L(\mathbf{w}) p(\mathbf{w})$$

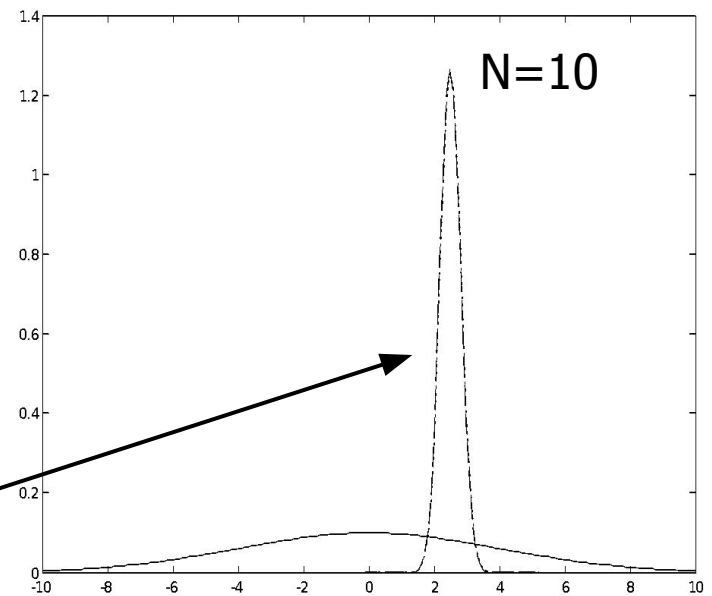
If prior is very 'weak' (i.e., σ_0^2 is large) then likelihood dominates.

Typically as $N \rightarrow \infty$

$$p(\mathbf{w} | \chi) \rightarrow L(\mathbf{w})$$

Also, as $N \rightarrow \infty$, $L(\mathbf{w})$ becomes very narrow (variance $\rightarrow 0$), so the maximum value characterises the distribution well.

Likelihood \approx posterior



Summary

- o Introduced the notion of Learning probability models.
- o Maximum Likelihood method:
 - Solved through optimization
 - There is a danger of overfitting
- o Bayesian method:
 - Uses a distribution of parameters
 - Full Bayesian estimators involve integration
 - ML and Bayesian methods linked through likelihood function
- o Overfitting of models is a important practical concern