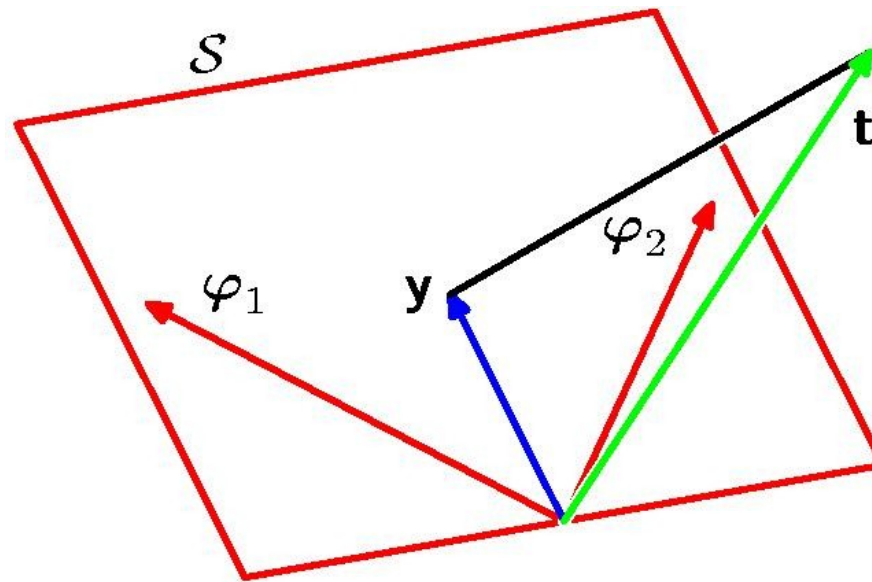
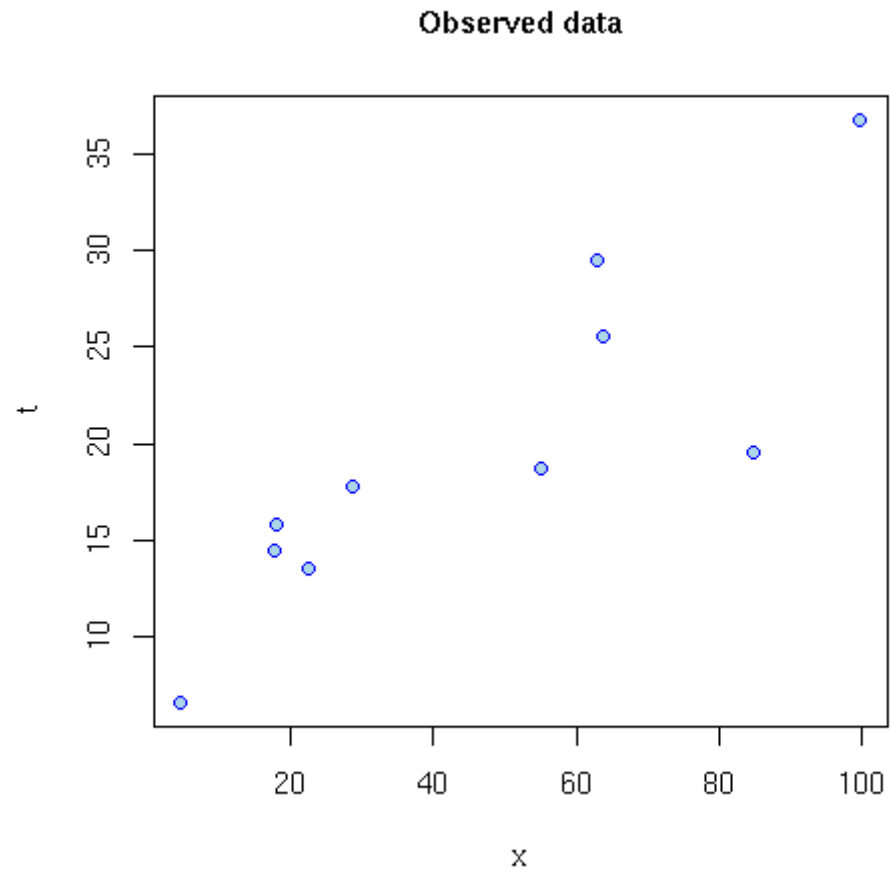


Linear regression

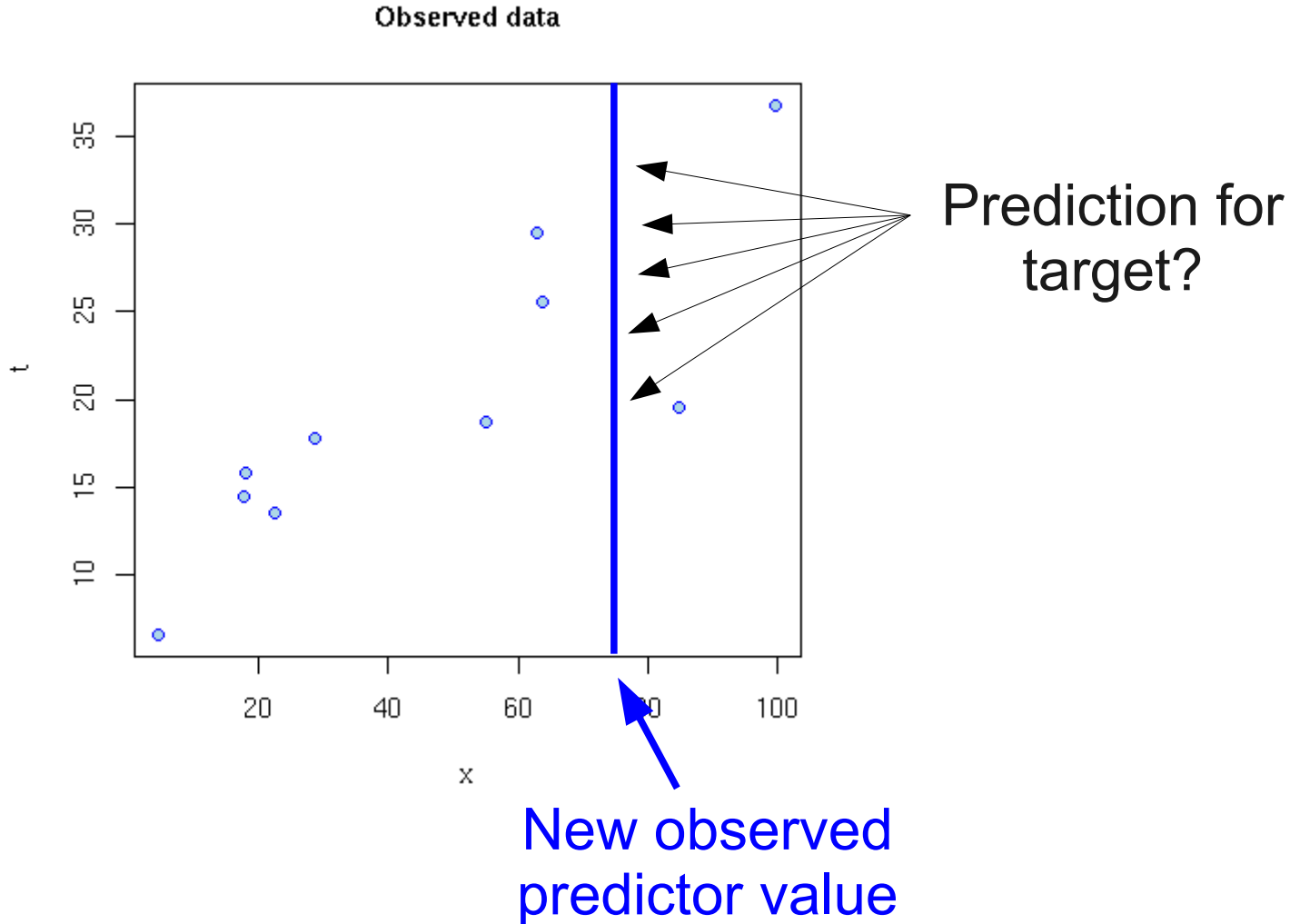


Machine Learning; Mon Apr 21, 2008

Motivation



Motivation



Motivation

Problem: We want a general way of obtaining a distribution $p(x,t)$ fitted to observed data.

If we don't try to interpret the distribution, then any distribution with non-zero value at the data points will do.

We will use theory from last week to construct **generic** approaches to learning distributions from data.

Motivation

Problem: We want a general way of obtaining a distribution $p(x,t)$ fitted to observed data.

If we don't try to interpret the distribution, then any distribution with non-zero value at the data points will do.

We will use theory from last week to construct **generic** approaches to learning distributions from data.

In this lecture: ***linear (normal/Gaussian) models.***

Linear Gaussian Models

In a linear Gaussian model, we model $p(\mathbf{x}, t)$ as a conditional Gaussian distribution

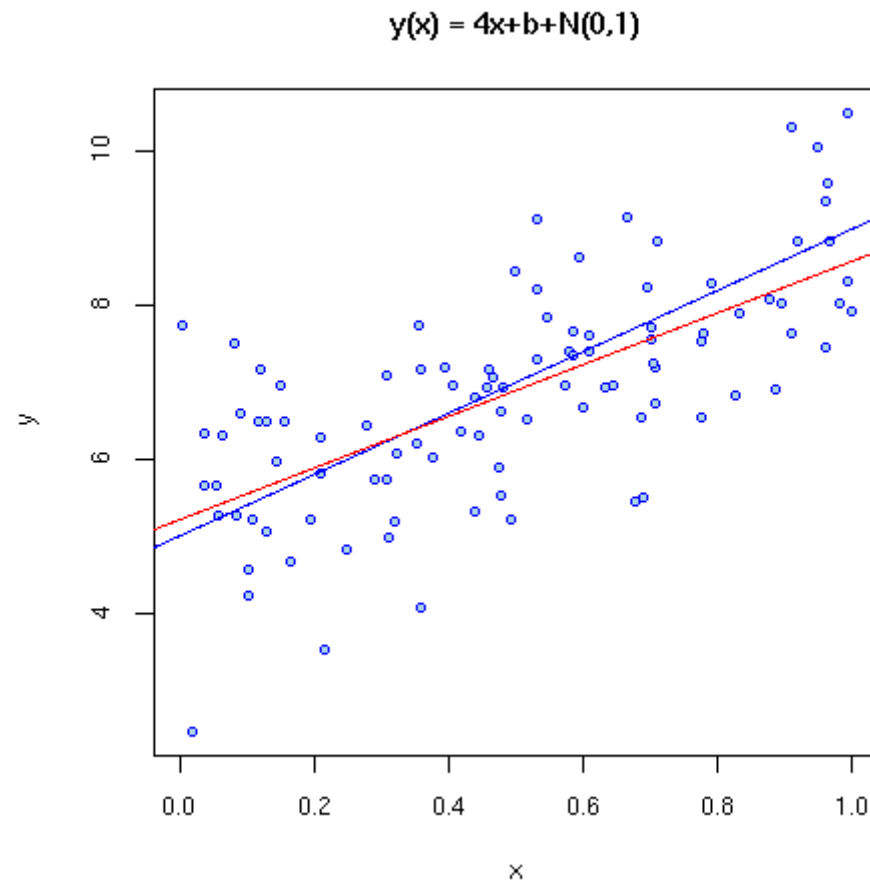
$$p(t \mid \mathbf{x}) \sim N(\mu_{\mathbf{x}}, \sigma^2)$$

where the x dependent mean depends linearly on a set of weights w .

$$\mu_{\mathbf{x}} = y(\mathbf{w}, \mathbf{x})$$

Example

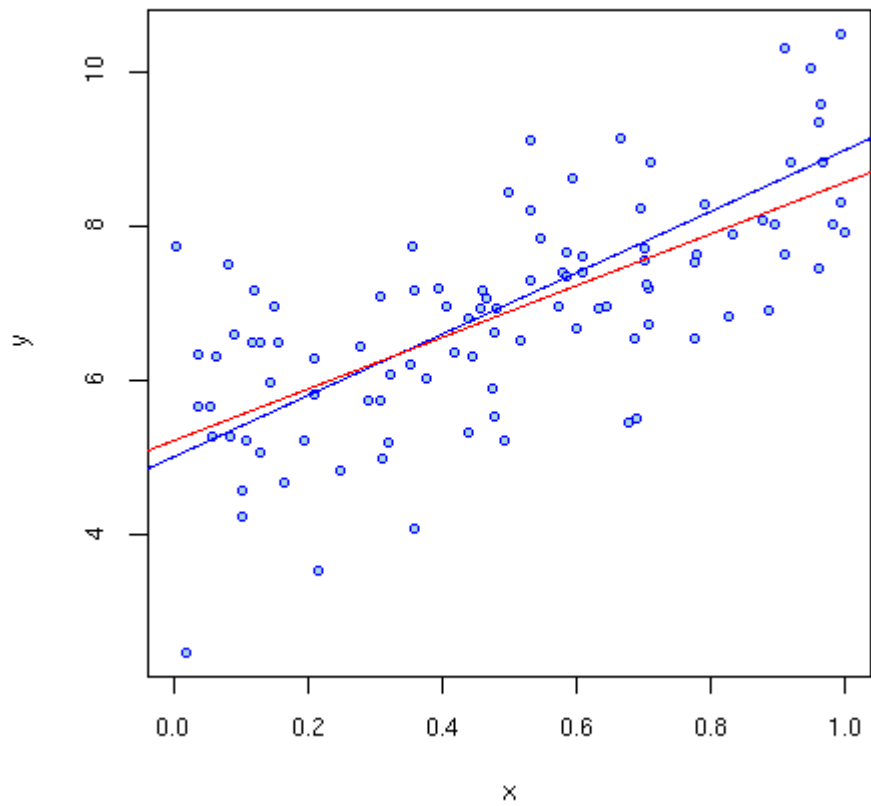
$$y(x, \mathbf{w}) = w_0 + w_1 \cdot x$$



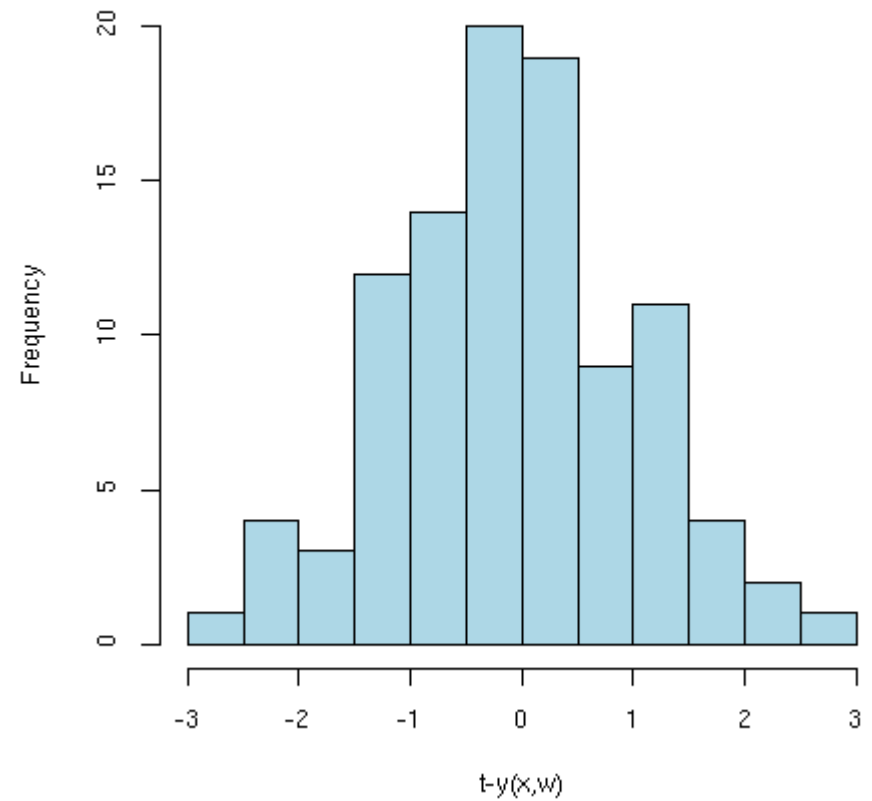
Example

$$y(x, \mathbf{w}) = w_0 + w_1 \cdot x$$

$y(x) = 4x + b + N(0,1)$



Histogram of residuals



General linear in input

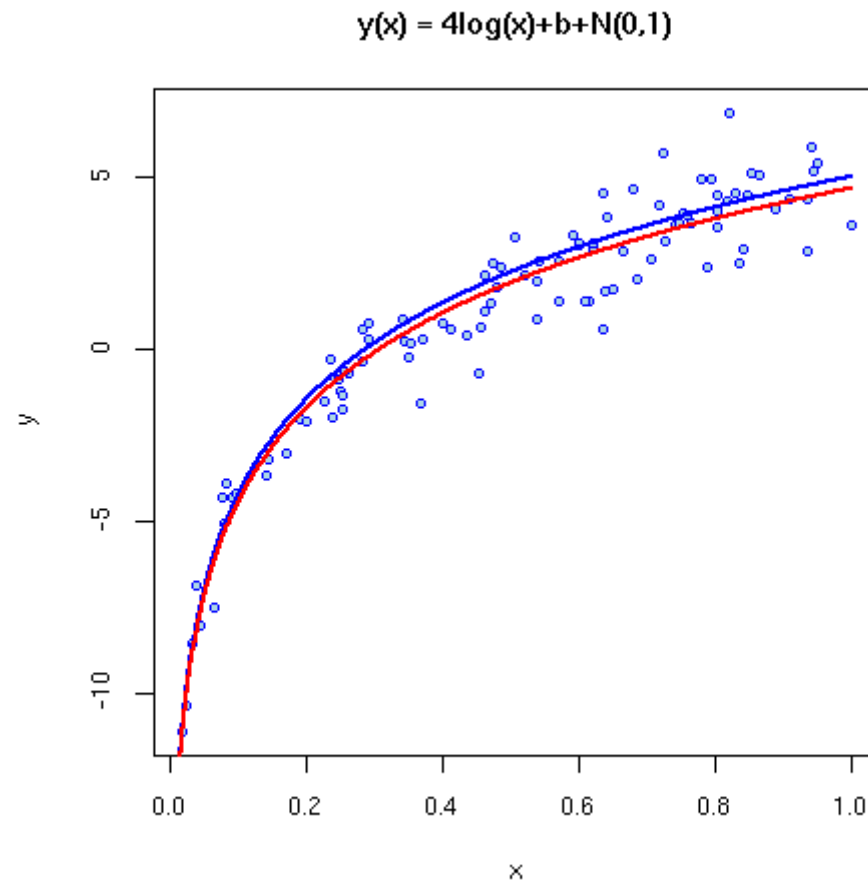
$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{i=1}^{M-1} w_i \cdot x_i$$

...or adding a pseudo input $x_0=1$

$$y(\mathbf{x}, \mathbf{w}) = \sum_{i=0}^{M-1} w_i \cdot x_i = \mathbf{w}^T \mathbf{x}$$

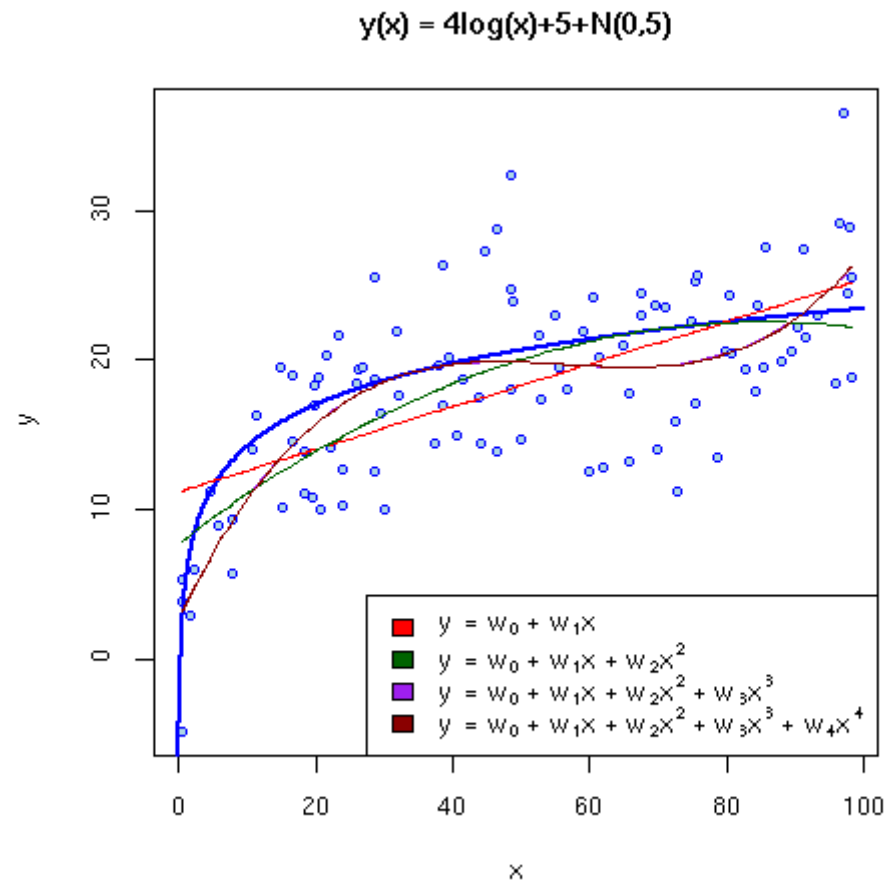
Non-linear in input (but still in weights)

$$y(x, \mathbf{w}) = w_0 + w_1 \cdot \log(x)$$



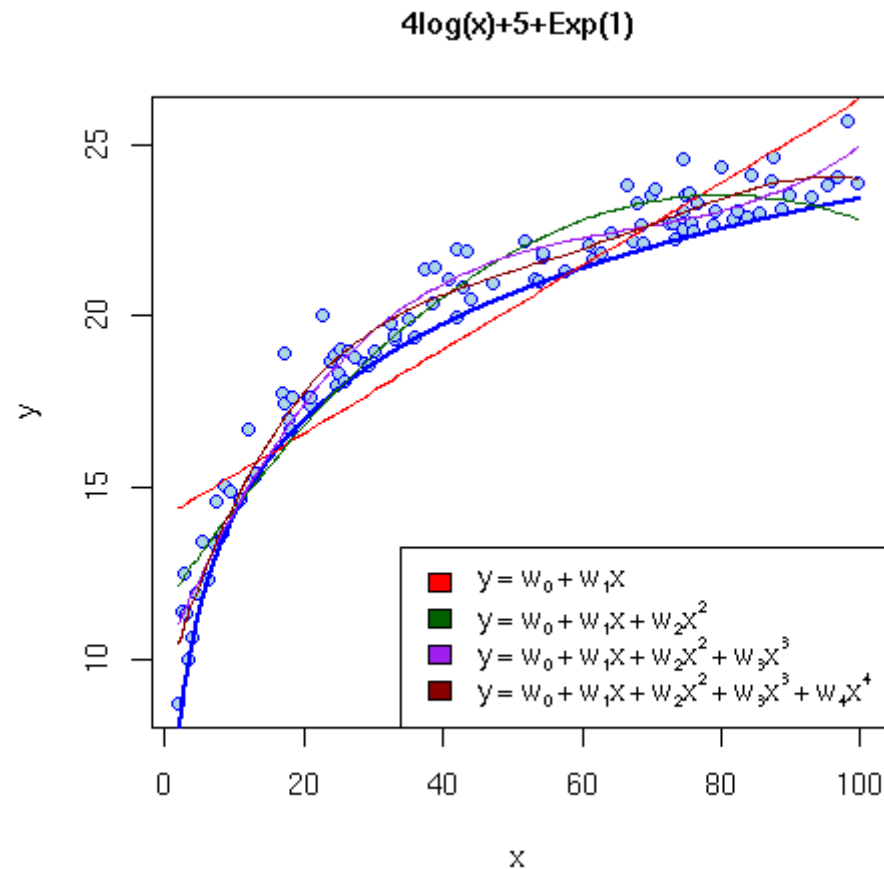
Non-linear in input (but still in weights)

But remember that we do not know the “true” underlying function...



Non-linear in input (but still in weights)

...nor the noise around the function...



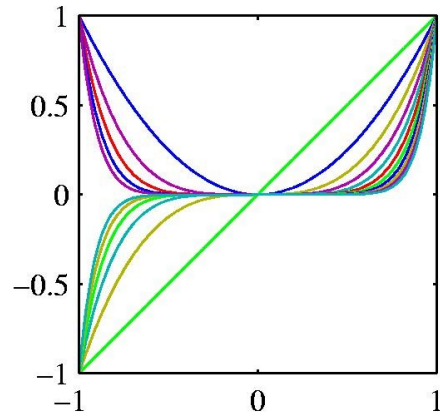
General linear model

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{i=1}^{M-1} w_i \cdot \phi_i(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$$

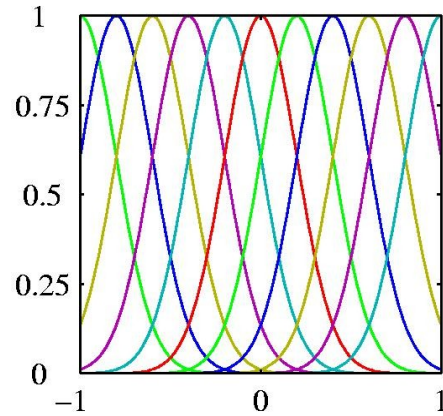
Basis functions.

Sometimes called “features”.

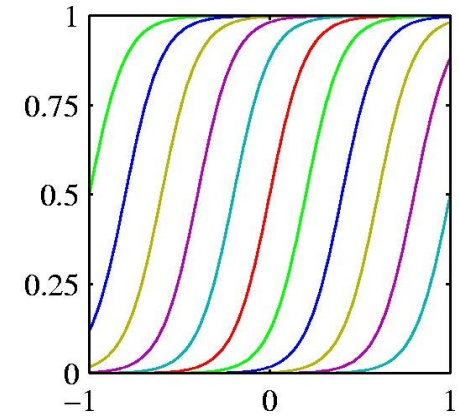
Examples of basis functions



Polynomials



Gaussians



Sigmoids

Estimating parameters

Observed data: $\mathcal{D} = \{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \dots, (\mathbf{x}_N, t_N)\}$

Log likelihood:

$$\begin{aligned}\log p(\mathcal{D} \mid \mathbf{w}, \sigma^2) &= \sum_{n=1}^N \log N(t_n \mid \mathbf{w}^T \phi(\mathbf{x}_n), \sigma^2) \\ &= \frac{N}{2} \log \left(\frac{1}{\sigma^2} \right) - \frac{N}{2} \log(2\pi) - \frac{1}{\sigma^2} E_{\mathcal{D}}(\mathbf{w})\end{aligned}$$

$$E_{\mathcal{D}}(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2$$

Estimating parameters

Ob Maximizing wrt \mathbf{w} means minimizing E – the **error function**.

Log likelihood:

$$\begin{aligned}\log p(\mathcal{D} \mid \mathbf{w}, \sigma^2) &= \sum_{n=1}^N \log N(t_n \mid \mathbf{w}^T \phi(\mathbf{x}_n), \sigma^2) \\ &= \frac{N}{2} \log \left(\frac{1}{\sigma^2} \right) - \frac{N}{2} \log(2\pi) - \frac{1}{\sigma^2} E_{\mathcal{D}}(\mathbf{w})\end{aligned}$$

$$E_{\mathcal{D}}(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2$$

Estimating parameters

$$E_{\mathcal{D}}(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2$$

$$\begin{aligned} \nabla E_{\mathcal{D}}(\mathbf{w}) &= \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\} \phi(\mathbf{x}_n)^T \\ &= \sum_{n=1}^N t_n \phi(\mathbf{x}_n)^T - \mathbf{w}^T \sum_{n=1}^N \phi(\mathbf{x}_n) \phi(\mathbf{x}_n)^T = 0 \end{aligned}$$

Estimating parameters

$$\hat{\mathbf{w}} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{t}$$

$$\hat{\sigma}^2 = \frac{1}{N} \sum_{n=1}^N (t_n - \hat{\mathbf{w}}^T \phi(\mathbf{x}_n))^2$$

Estimating parameters

$$\hat{\mathbf{w}} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{t}$$

$$\hat{\sigma}^2 = \frac{1}{N} \sum_{n=1}^N (t_n - \hat{\mathbf{w}}^T \phi(\mathbf{x}_n))^2$$

Notice: This is not just pure mathematics but an actual algorithm for estimating (learning) the parameters!

C with GSL and CBLAS

```
emacs21@localhost
#include <stdio.h>
#include <stdlib.h>
#include <gsl/gsl_linalg.h>
#include <gsl/gsl_blas.h>

int main(int argc, char *argv[])
{
    int r = 3;
    int c = 2;
    double phi_data[] = { 1.0, 1.0,
                          1.0, 2.0,
                          1.0, 3.0 };

    double t_data[] = { 2.1,
                       3.8,
                       6.2 };

    /* phi and target vector */
    gsl_matrix_view phi = gsl_matrix_view_array(phi_data, r, c);
    gsl_vector_view t = gsl_vector_view_array(t_data, r);

    /* CBLAS matrix multiplication: phiTphi <- 1 * phi^T * phi + 0 */
    gsl_matrix *phiTphi = gsl_matrix_alloc(c, c);
    gsl_blas_dgemm(CblasTrans, CblasNoTrans,
                  1, &phi.matrix, &phi.matrix, 0, phiTphi);

    /* compute inverse using LU decomposition */
    gsl_matrix *phiTphiInv = gsl_matrix_alloc(c, c);
    gsl_permutation *perm = gsl_permutation_alloc(c);
    int s = 0;
    gsl_linalg_LU_decomp(phiTphi, perm, &s); /* phiTphi now LU decomposed */
    gsl_linalg_LU_invert(phiTphi, perm, phiTphiInv);

    /* compute w as phiTphiInv * phi^T * t */
    gsl_vector *phit = gsl_vector_alloc(c);
    gsl_vector *w = gsl_vector_alloc(c);
    /* phit <- phi^T * t */
    gsl_blas_dgemv(CblasTrans, 1, &phi.matrix, &t.vector, 0, phit);
    /* w <- phiTphiInv * phit */
    gsl_blas_dgemv(CblasNoTrans, 1, phiTphiInv, phit, 0, w);

    int i;
    for (i = 0; i < c; ++i)
        printf("%5.3g\n", gsl_vector_get(w, i));

    gsl_matrix_free(phiTphi);
    gsl_matrix_free(phiTphiInv);
    gsl_vector_free(phit);
    gsl_vector_free(w);

    return EXIT_SUCCESS;
}
```

Notice
actual
para

Octave/ Matlab

```
emacs21@localhost
#include <stdio.h>
#include <stdlib.h>
#include <gsl/gsl_linalg.h>
#include <gsl/gsl_blas.h>

int main(int argc, char *argv[])
{
    int r = 3;
    int c = 2;
    double phi_data[] = { 1.0, 1.0,
                          1.0, 2.0,
                          1.0, 3.0 };

    double t_data[] = { 2.1,
                       3.8,
                       6.2 };

    /* phi and target vector */
    gsl_matrix_view phi = gsl_matrix_view_array(phi_data, r, c);
    gsl_vector view t = gsl_vector_view_array(t_data, r);
```

```
emacs21@localhost
phi = [1.0 1.0
       1.0 2.0
       1.0 3.0];
t = [2.1
     3.8
     6.2]

w = inv(phi' * phi) * phi' * t

-u:-- narko.m (Octave)--L10--All-----
```

Notice
actual
para

```
/* phit <- phi' * t */
gsl_blas_dgemv(CblasTrans, 1, &phi.matrix, &t.vector, 0, phit);
/* w <- phiTphiInv * phit */
gsl_blas_dgemv(CblasNoTrans, 1, phiTphiInv, phit, 0, w);

int i;
for (i = 0; i < c; ++i)
    printf("%5.3g\n", gsl_vector_get(w, i));

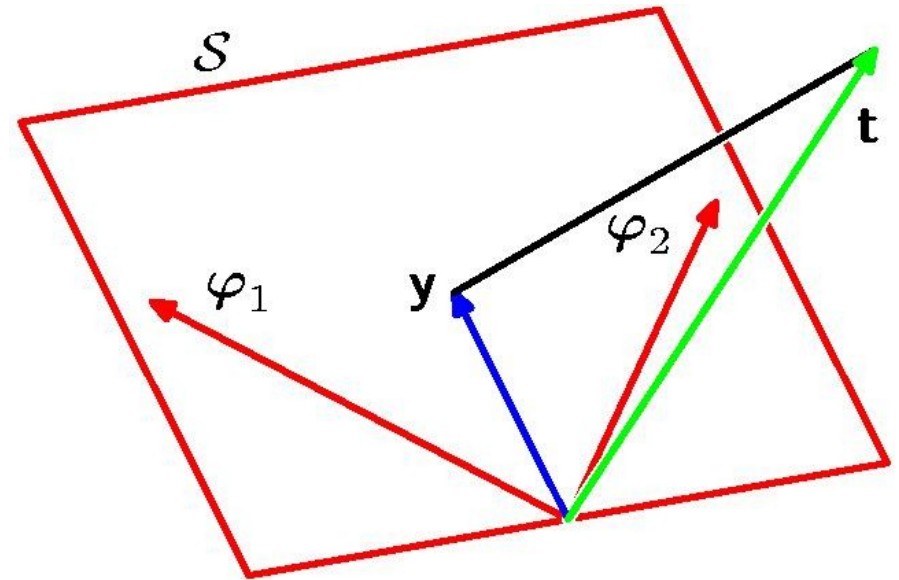
gsl_matrix_free(phiTphi);
gsl_matrix_free(phiTphiInv);
gsl_vector_free(phit);
gsl_vector_free(w);

return EXIT_SUCCESS;
}

-u:-- narko.c (C Abbrev)--L53--All-----
```

Geometrical interpretation

Geometrically y is the projection of t onto the space spanned by the features:



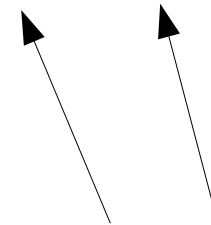
Bayesian linear regression

For the Bayesian approach we need a prior over the parameters w and $\beta = 1/\sigma^2$

Conjugate for Gaussian is Gaussian:

$$p(\mathbf{w}) = N(\mathbf{w} \mid \mathbf{m}_0, \mathbf{S}_0)$$

$$p(\mathbf{w} \mid \mathbf{x}, \mathbf{t}) = N(\mathbf{w} \mid \mathbf{m}_N, \mathbf{S}_N)$$



Functions of observed values

Bayesian linear regression

For the Bayesian approach we need a prior over the parameters w and $\beta = 1/\sigma^2$

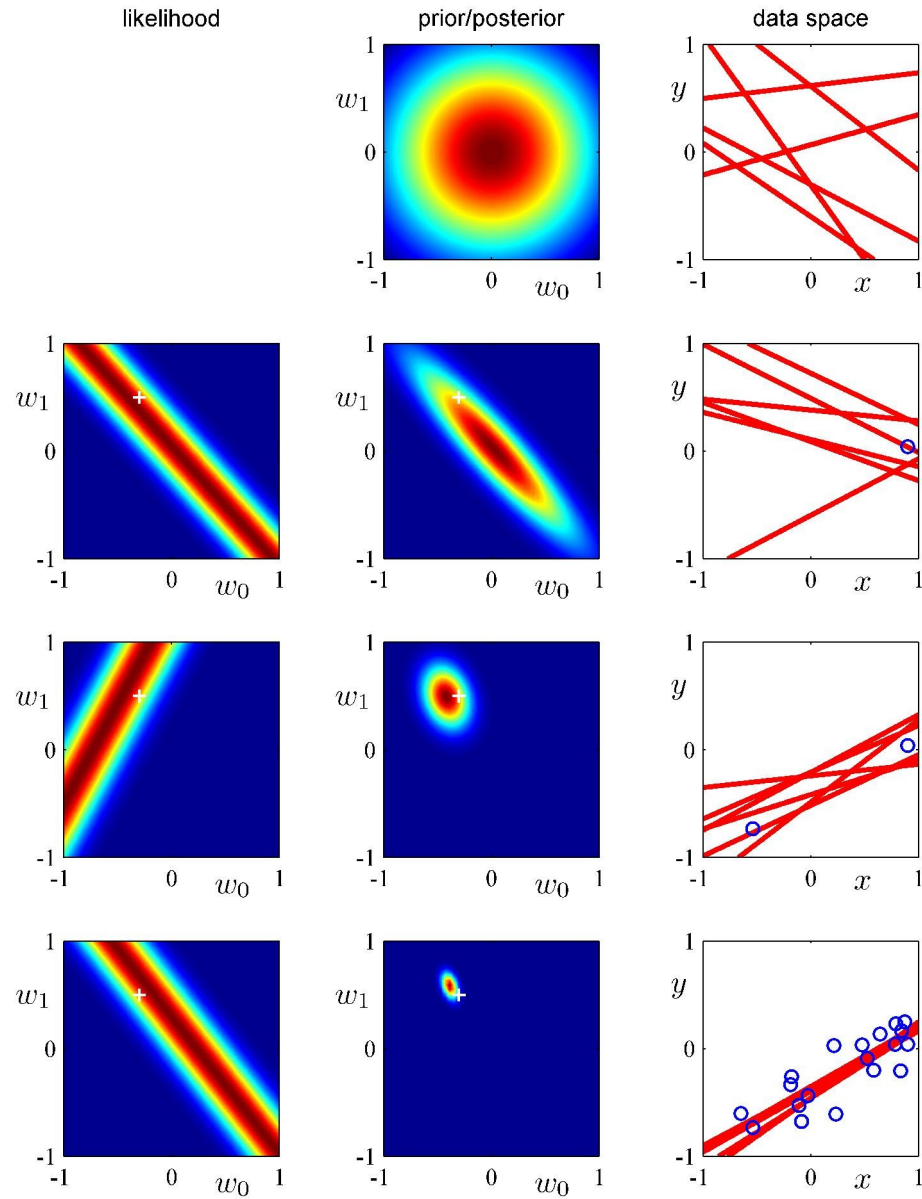
Conjugate for Gaussian is Gaussian:

$$p(\mathbf{w}) = N(\mathbf{w} \mid \mathbf{m}_0, \mathbf{S}_0)$$

$$p(\mathbf{w} \mid \mathbf{x}, \mathbf{t}) = N(\mathbf{w} \mid \mathbf{m}_N, \mathbf{S}_N)$$

Proof not *exactly* like before, but similar, and uses linearity results from Gaussians' from 2.3.3.

Example



Bayesian linear regression

Predictor for future observations is also Gaussian (again result from 2.3.3):

$$p(t \mid \mathbf{y}, \mathbf{x}, \mathbf{t}) = \int p(t \mid \mathbf{y}, \mathbf{w})p(\mathbf{w} \mid \mathbf{x}, \mathbf{t}) d\mathbf{w}$$

Bayesian linear regression

Predictor for future observations is also Gaussian (again result from 2.3.3):

$$\begin{aligned} p(t \mid \mathbf{y}, \mathbf{x}, \mathbf{t}) &= \int p(t \mid \mathbf{y}, \mathbf{w}) p(\mathbf{w} \mid \mathbf{x}, \mathbf{t}) d\mathbf{w} \\ &= N(t \mid \mathbf{m}_{\mathbf{x}, \mathbf{t}}^T \phi(\mathbf{y}), \sigma_{\mathbf{x}, \mathbf{t}}^2(\mathbf{y})) \end{aligned}$$

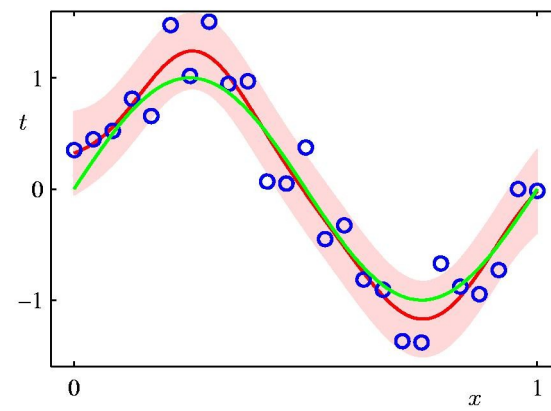
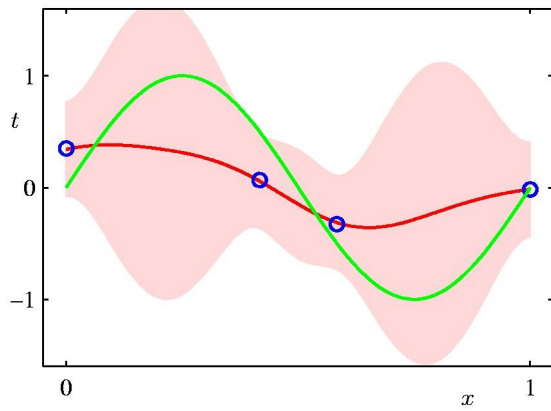
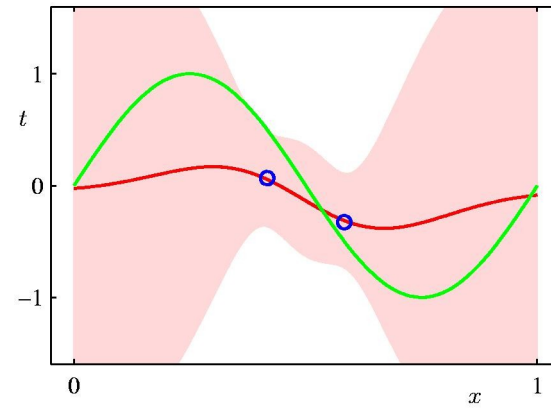
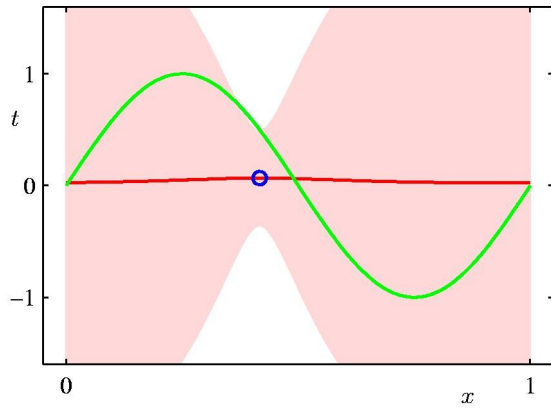
Bayesian linear regression

Predictor for future observations is also Gaussian (again result from 2.3.3):

$$\begin{aligned} p(t \mid \mathbf{y}, \mathbf{x}, \mathbf{t}) &= \int p(t \mid \mathbf{y}, \mathbf{w}) p(\mathbf{w} \mid \mathbf{x}, \mathbf{t}) d\mathbf{w} \\ &= N(t \mid \mathbf{m}_{\mathbf{x}, \mathbf{t}}^T \phi(\mathbf{y}), \sigma_{\mathbf{x}, \mathbf{t}}^2(\mathbf{y})) \end{aligned}$$

Both mean and variance of this distribution depends on \mathbf{y} !

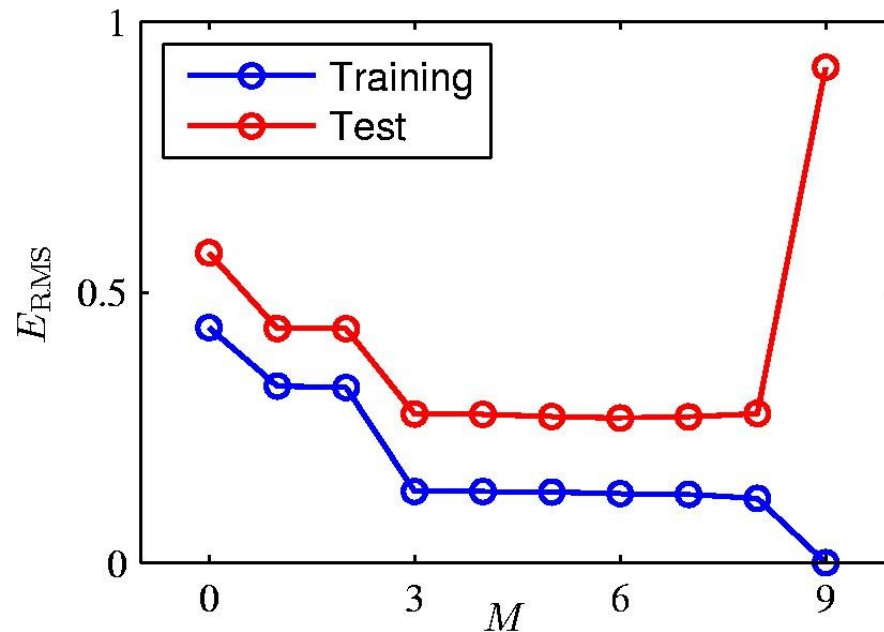
Example



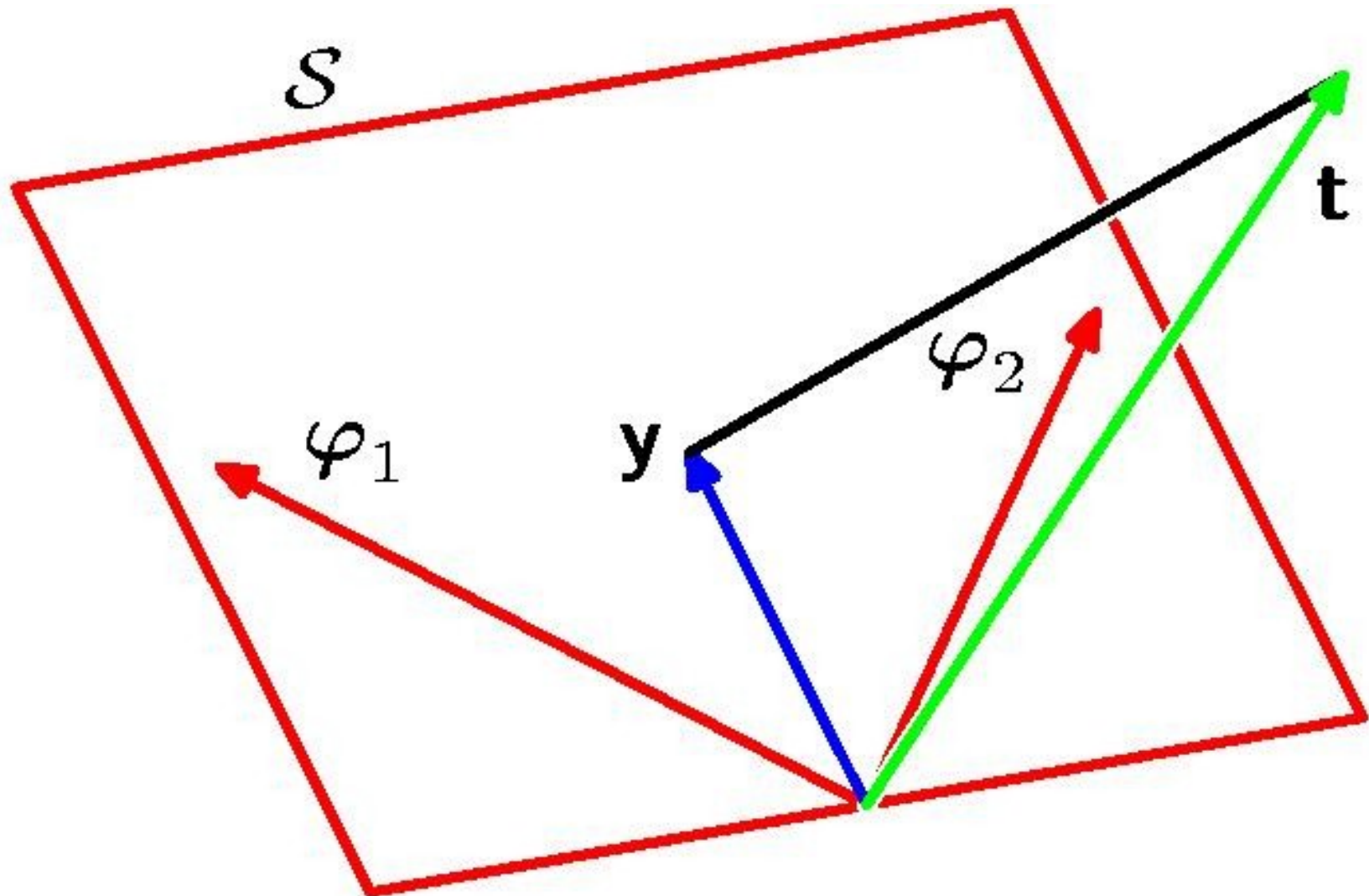
Over fitting

Problem: Over-fitting is always a problem when we fit data to generic models.

With nested models, the ML parameters will *never* prefer a simple model over a more complex model...



Maximum likelihood problems



Bayesian model selection

We can take a more Bayesian approach and select model based on posterior model probabilities:

$$p(\mathcal{M}_i | \mathcal{D}) \propto p(\mathcal{D} | \mathcal{M}_i)p(\mathcal{M}_i)$$

Bayesian model selection

We can take a more Bayesian approach and select model based on posterior model probabilities:

$$p(\mathcal{M}_i | \mathcal{D}) \propto p(\mathcal{D} | \mathcal{M}_i)p(\mathcal{M}_i)$$

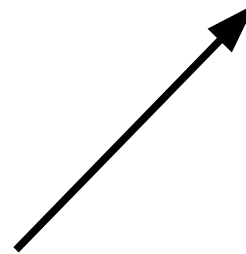
The normalizing factor is the same for all models:

$$p(\mathcal{M}_i | \mathcal{D}) = \frac{p(\mathcal{D} | \mathcal{M}_i)p(\mathcal{M}_i)}{p(\mathcal{D})}$$

Bayesian model selection

We can take a more Bayesian approach and select model based on posterior model probabilities:

$$p(\mathcal{M}_i | \mathcal{D}) \propto p(\mathcal{D} | \mathcal{M}_i)p(\mathcal{M}_i)$$

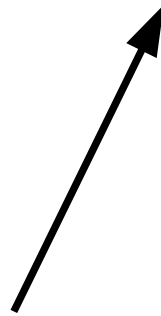


The prior captures our preferences in the models.

Bayesian model selection

We can take a more Bayesian approach and select model based on posterior model probabilities:

$$p(\mathcal{M}_i | \mathcal{D}) \propto p(\mathcal{D} | \mathcal{M}_i)p(\mathcal{M}_i)$$



The likelihood captures the data's preferences in models.

The marginal likelihood

The likelihood of the model is the integral over all the models parameters:

$$p(\mathcal{D} \mid \mathcal{M}_i) = \int p(\mathcal{D} \mid \mathbf{w}, \mathcal{M}_i) p(\mathbf{w} \mid \mathcal{M}_i) d\mathbf{w}$$

The marginal likelihood

The likelihood of the model is the integral over all the models parameters:

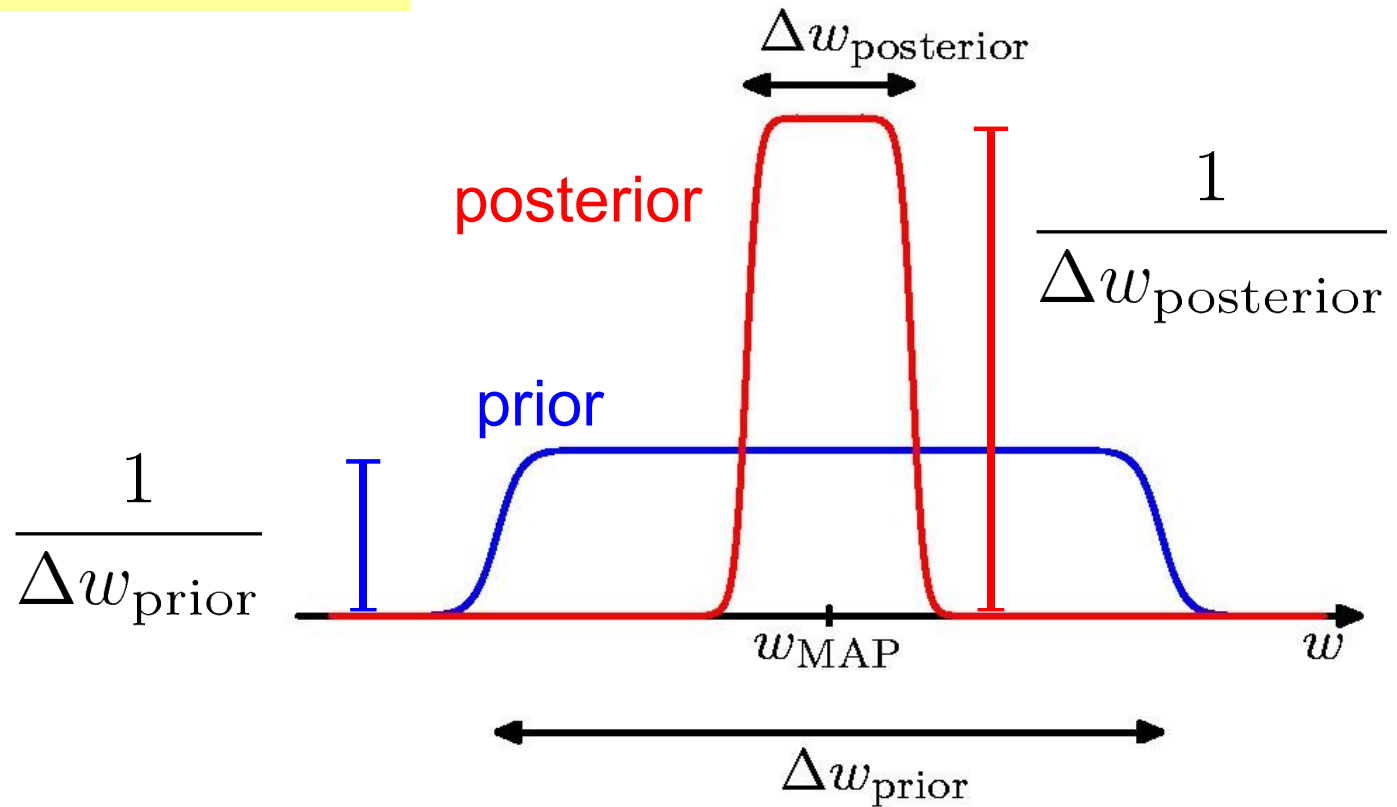
$$p(\mathcal{D} | \mathcal{M}_i) = \int p(\mathcal{D} | \mathbf{w}, \mathcal{M}_i) p(\mathbf{w} | \mathcal{M}_i) d\mathbf{w}$$

which is also the normalizing factor for the posterior:

$$p(\mathbf{w} | \mathcal{D}, \mathcal{M}_i) = \frac{p(\mathcal{D} | \mathbf{w}, \mathcal{M}_i) p(\mathbf{w} | \mathcal{M}_i)}{p(\mathcal{D} | \mathcal{M}_i)}$$

Implicit over-fitting penalty

Assume this is the shape of prior and posterior

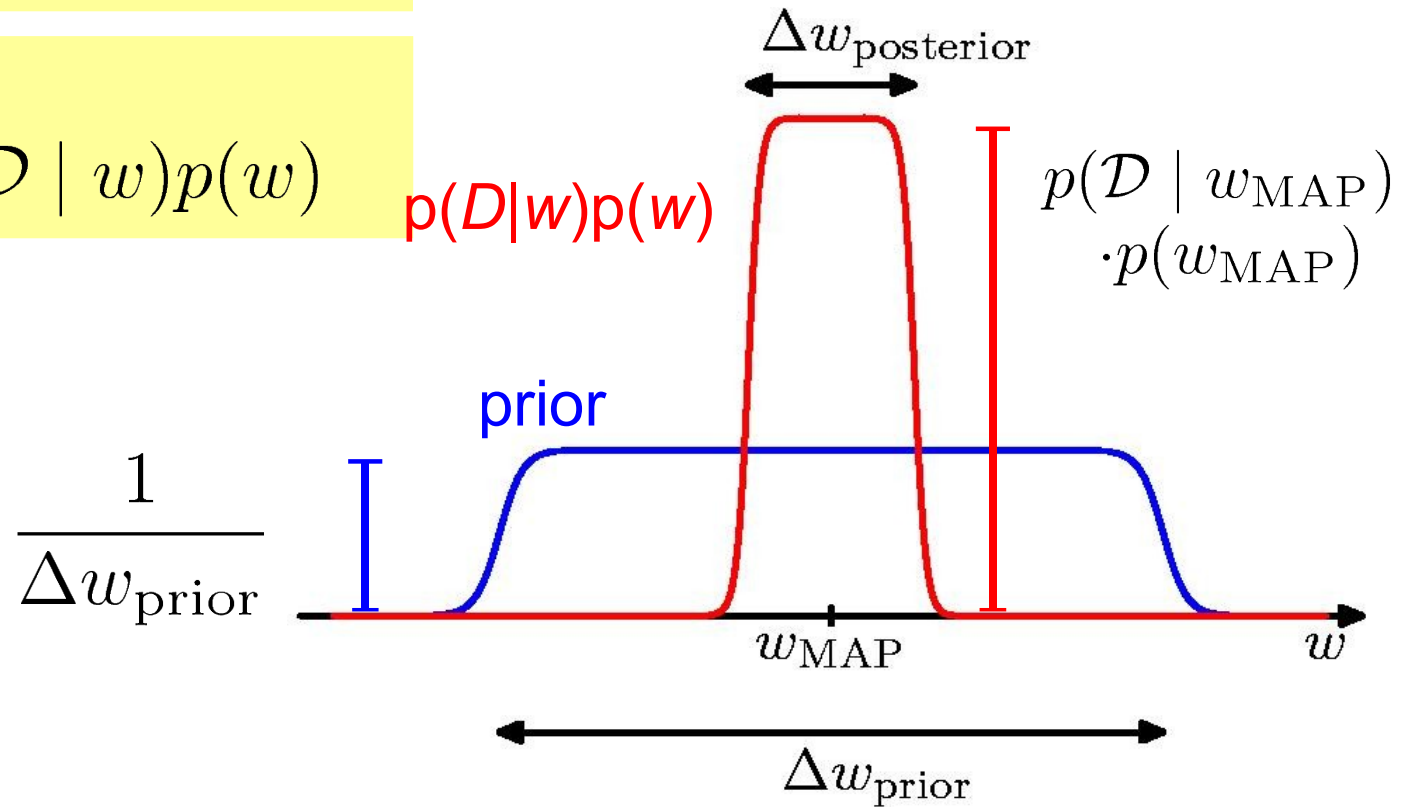


Implicit over-fitting penalty

Assume this is the shape of prior and posterior

By proportionality

$$p(w | \mathcal{D}) \propto p(\mathcal{D} | w)p(w)$$



Implicit over-fitting penalty

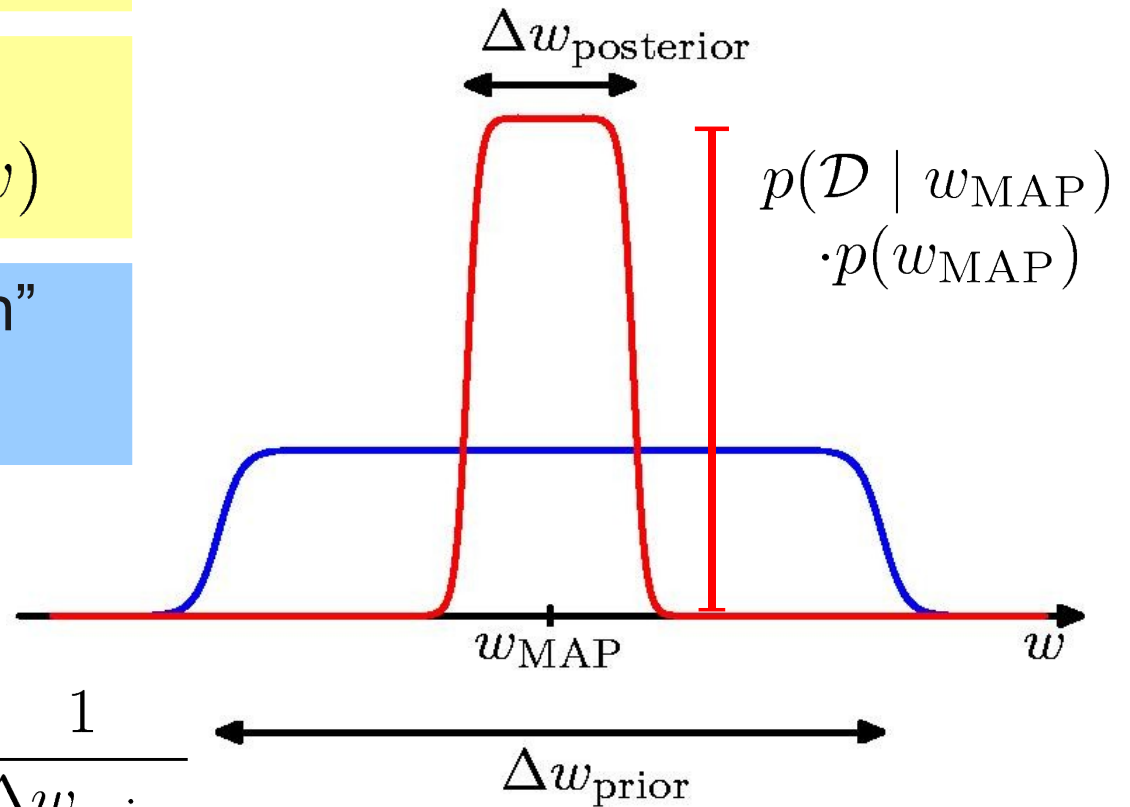
Assume this is the shape of prior and posterior

By proportionality

$$p(w \mid \mathcal{D}) \propto p(\mathcal{D} \mid w)p(w)$$

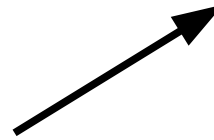
Integral approximately “width” times “height”

$$\int p(\mathcal{D} \mid w)p(w) dw \approx \Delta w_{\text{posterior}} \cdot p(\mathcal{D} \mid w_{\text{MAP}}) \cdot \frac{1}{\Delta w_{\text{prior}}}$$



Implicit over-fitting penalty

$$\log p(\mathcal{D}) \approx \log p(\mathcal{D} \mid w_{\text{MAP}}) + \log \left(\frac{\Delta w_{\text{posterior}}}{\Delta w_{\text{prior}}} \right)$$



Increasingly negative as posterior becomes “pointy” compared to prior

Close fitting to data is implicitly penalized, and the marginal likelihood is a trade-off between maximizing the posterior and minimizing this penalty.

Implicit over-fitting penalty

$$\log p(\mathcal{D}) \approx \log p(\mathcal{D} \mid w_{\text{MAP}}) + M \log \left(\frac{\Delta w_{\text{posterior}}}{\Delta w_{\text{prior}}} \right)$$



Penalty increases with number of parameters M

Close fitting to data is implicitly penalized, and the marginal likelihood is a trade-off between maximizing the posterior and minimizing this penalty.

On average we prefer the true model

This doesn't mean we always prefer the simplest model!

One can show

$$\int p(\mathcal{D} | \mathcal{M}_1) \ln \frac{p(\mathcal{D} | \mathcal{M}_1)}{p(\mathcal{D} | \mathcal{M}_2)} d\mathcal{D} \geq 0$$

with zero only when

$$\mathcal{M}_1 = \mathcal{M}_2$$

i.e. on average the right model is the preferred model.

On average we prefer the true model

This doesn't mean we always prefer the simplest model!

One can show

$$\int p(\mathcal{D} | \mathcal{M}_1) \ln \frac{p(\mathcal{D} | \mathcal{M}_1)}{p(\mathcal{D} | \mathcal{M}_2)} d\mathcal{D} \geq 0$$

with zero only when

$$\mathcal{M}_1 = \mathcal{M}_2$$

i.e. on average the right model is the preferred model.

Negative when we prefer the second model
positive when we prefer the first

On average we prefer the true model

This doesn't mean we always prefer the simplest model!

One can show

$$\int p(\mathcal{D} | \mathcal{M}_1) \ln \frac{p(\mathcal{D} | \mathcal{M}_1)}{p(\mathcal{D} | \mathcal{M}_2)} d\mathcal{D} \geq 0$$

with zero only when

$$\mathcal{M}_1 = \mathcal{M}_2$$

i.e. on average the right model is the preferred model.



On average, we will not prefer the second model when the first is true...

On average we prefer the true model

This doesn't mean we always prefer the simplest model!

One can show

$$\int p(\mathcal{D} | \mathcal{M}_1) \ln \frac{p(\mathcal{D} | \mathcal{M}_1)}{p(\mathcal{D} | \mathcal{M}_2)} d\mathcal{D} \geq 0$$

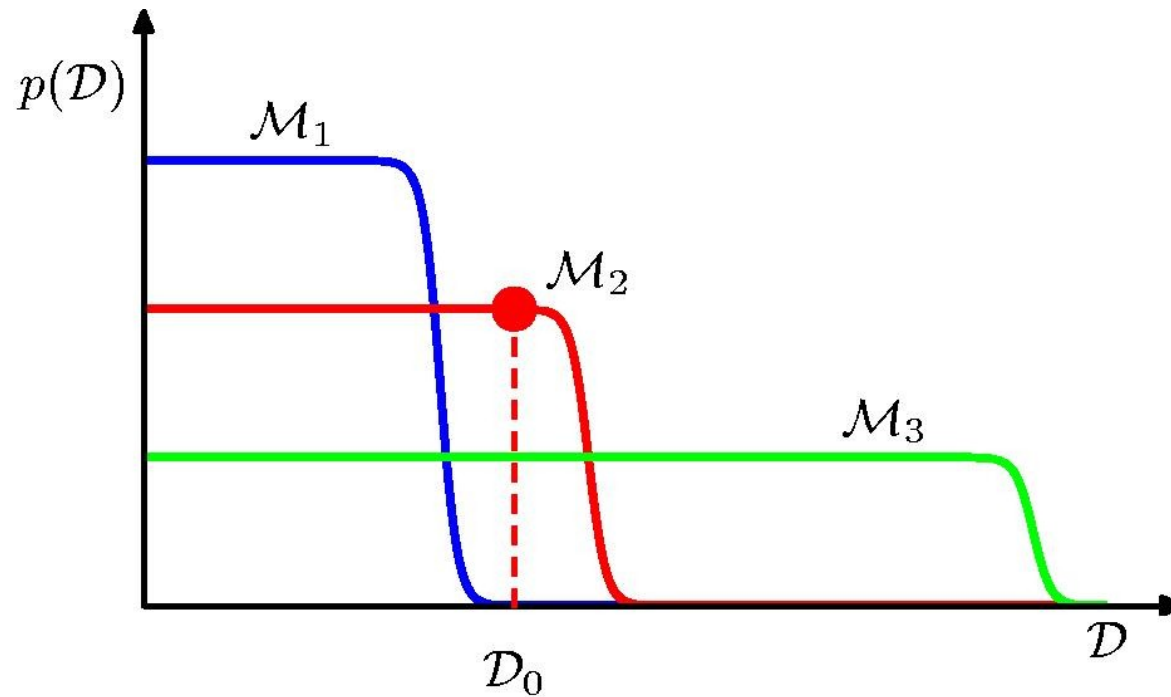
with zero only when

$$\mathcal{M}_1 = \mathcal{M}_2$$

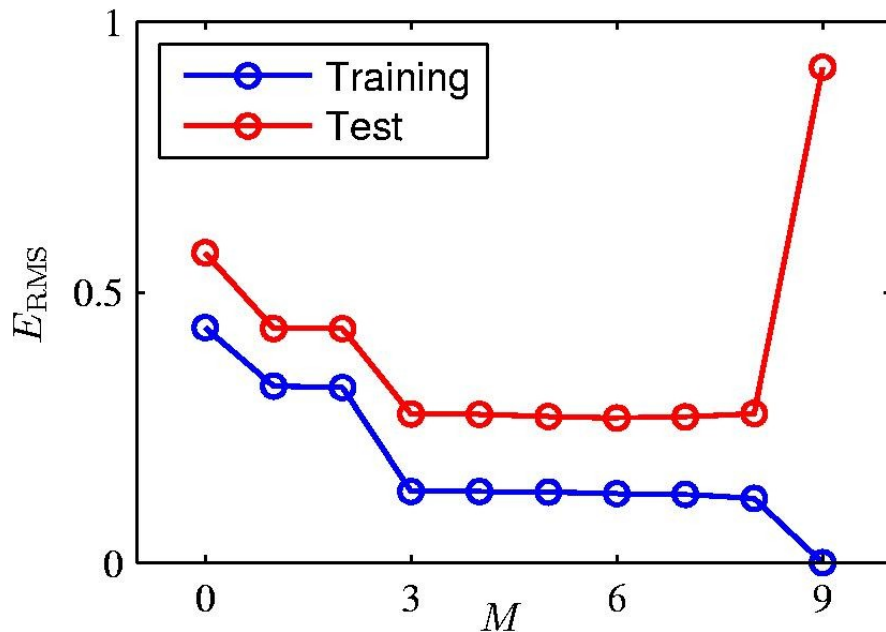
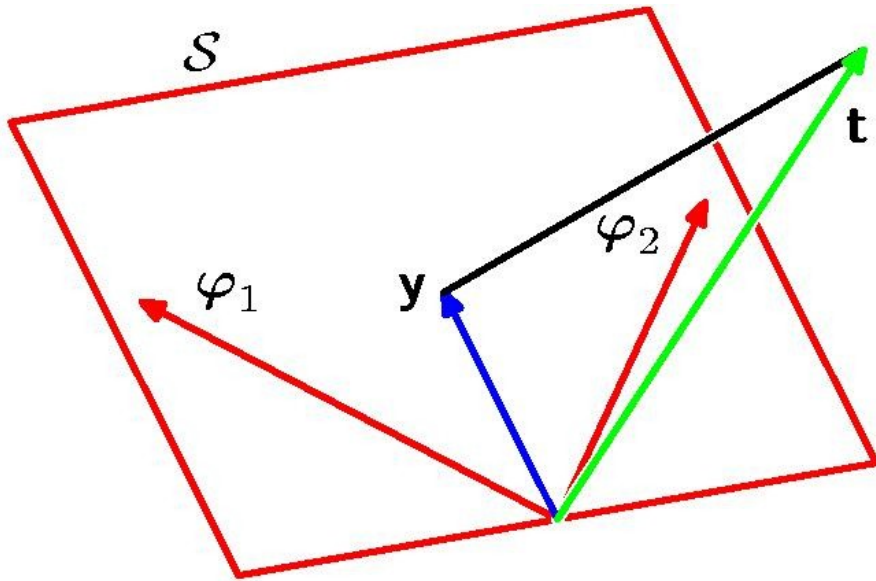
i.e. on average the right model is the preferred model.

Close fitting to data is implicitly penalized, and the marginal likelihood is a trade-off between maximizing the posterior and minimizing this penalty.

**On average we prefer
the true model**



Summary



- Linear Gaussians as generic densities
- ML or Bayesian estimation for training
- Over-fitting is an inherent problem in ML estimation
- Bayesian methods avoid the maximization caused over-fitting problem
 - (but is still vulnerable to model mis-specification)