Lecture 1 and 2: Probabilistic Regression

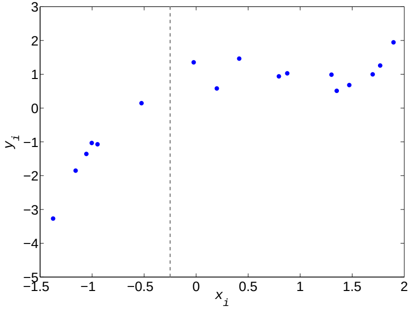
4F13: Machine Learning

Carl Edward Rasmussen and Zoubin Ghahramani

Department of Engineering
University of Cambridge

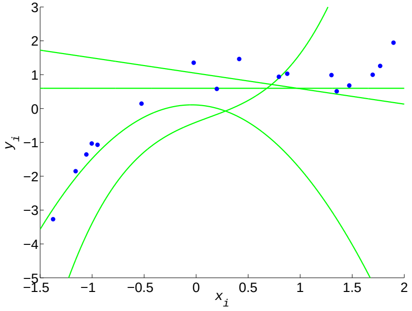
`http://mlg.eng.cam.ac.uk/teaching/4f13/`

How do we fit this dataset?



- Dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ of N pairs of inputs x_i and targets y_i.
  This data can for example be measurements in an experiment.
- Goal: predict target y_* associated to any arbitrary input x_*.
  This is known a as a regression task in machine learning.
- Note: Here the inputs are scalars, we have a single input feature.
  Inputs to regression tasks are often vectors of multiple input features.

Model of the data

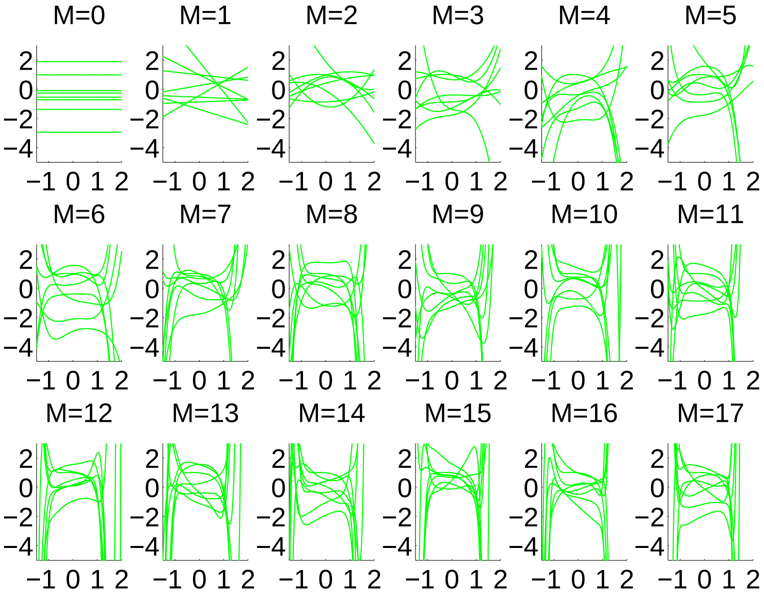


- In order to predict at a new x_* we need to postulate a model of the data. We will estimate y_* with $f(x_*)$.
- But what is $f(x)$? Example: a polynomial

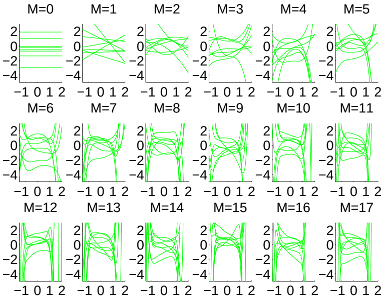
$$f_{\mathbf{w}}(x) \; = \; w_0 + w_1\, x + w_2\, x^2 + w_3\, x^3 + \ldots + w_M\, x^M$$

The w_j are the weights of the polynomial, the parameters of the model.

Model of the data. Example: polynomials of degree M

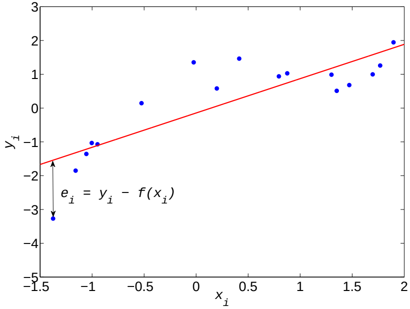


Model structure and model parameters



- Should we choose a polynomial? model structure
- What degree should we choose for the polynomial? model structure
- For a given degree, how do we choose the weights? model parameters
- For now, let find the single “best” polynomial: degree and weights.

Fitting model parameters: the least squares approach



- Idea: measure the quality of the fit to the training data.
- For each training point, measure the squared error $e_i^2 = (y_i - f(x_i))^2$.
- Find the parameters that minimise the sum of squared errors:

$$E(\mathbf{w}) = \sum_{i=1}^{N} e_i^2$$

$f_{\mathbf{w}}(x)$ is a function of the parameter vector $\mathbf{w} = [w_0, w_1, \ldots, w_M]^\top$.

Least squares in detail. (1) Notation

Some notation: training targets $\mathbf{y}$, predictions $\mathbf{f}$ and errors $\mathbf{e}$.

- $\mathbf{y} = [y_1, \ldots, y_N]^\top$ is a vector that stacks the N training targets.
- $\mathbf{f} = [f_{\mathbf{w}}(x_1), \ldots, f_{\mathbf{w}}(x_N)]^\top$ stacks $f_{\mathbf{w}}(x)$ evaluated at the N training inputs.
- $\mathbf{e} = \mathbf{y} - \mathbf{f}$ is the vector of training prediction errors.

The sum of squared errors is therefore given by

$$E(\mathbf{w}) = \|\mathbf{e}\|^2 = \mathbf{e}^\top \mathbf{e} = (\mathbf{y} - \mathbf{f})^\top (\mathbf{y} - \mathbf{f})$$

More notation: weights $\mathbf{w}$, basis functions $\phi_j(x)$ and matrix $\mathbf{\Phi}$.

- $\mathbf{w} = [w_0, w_1, \ldots, w_M]^\top$ stacks the $M + 1$ model weights.
- $\phi_j(x) = x^j$ is a basis function of our linear in the parameters model.

$$f_{\mathbf{w}}(x) = w_0\, 1 + w_1\, x + w_2\, x^2 + \ldots + w_M\, x^M = \sum_{j=0}^{M} w_j\, \phi_j(x)$$

- $\mathbf{\Phi}_{ij} = \phi_j(x_i)$ allows us to write $\mathbf{f} = \mathbf{\Phi}\, \mathbf{w}$.

Least squares in detail. (2) Solution

A Gradient View. The sum of squared errors is a convex function of $\mathbf{w}$:

$$\mathsf{E}(\mathbf{w}) \;=\; (\mathbf{y}-\mathbf{f})^\top(\mathbf{y}-\mathbf{f}) \;=\; (\mathbf{y}-\boldsymbol{\Phi}\,\mathbf{w})^\top(\mathbf{y}-\boldsymbol{\Phi}\,\mathbf{w})$$

The gradient with respect to the weights is:

$$\frac{\partial \mathsf{E}(\mathbf{w})}{\partial \mathbf{w}} \;=\; 2\,\boldsymbol{\Phi}^\top(\mathbf{y}-\boldsymbol{\Phi}\,\mathbf{w}) \;=\; 2\,\boldsymbol{\Phi}^\top\,\mathbf{y} - 2\boldsymbol{\Phi}^\top\,\boldsymbol{\Phi}\,\mathbf{w}$$

The weight vector $\hat{\mathbf{w}}$ that sets the gradient to zero minimises $\mathsf{E}(\mathbf{w})$:

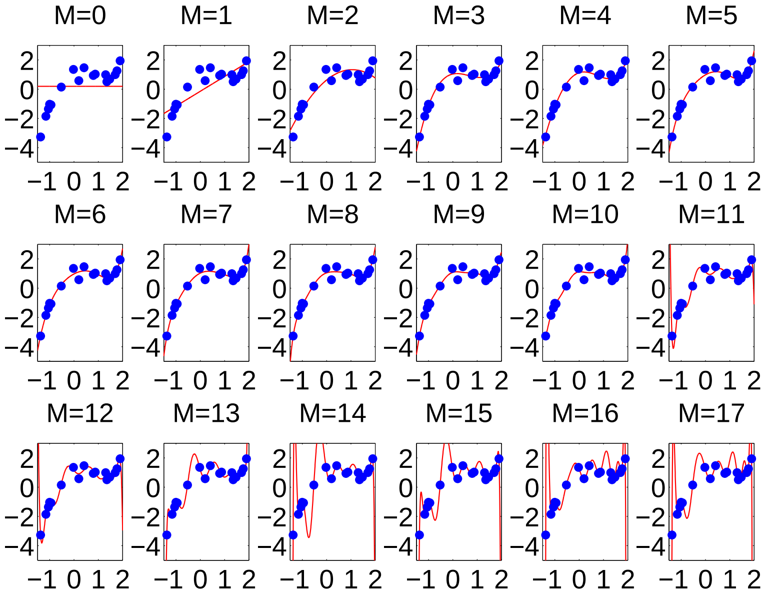
$$\boxed{\hat{\mathbf{w}} \;=\; (\boldsymbol{\Phi}^\top\,\boldsymbol{\Phi})^{-1}\,\boldsymbol{\Phi}^\top\,\mathbf{y}}$$

A Geometrical View. This is the matrix form of the Normal equations.

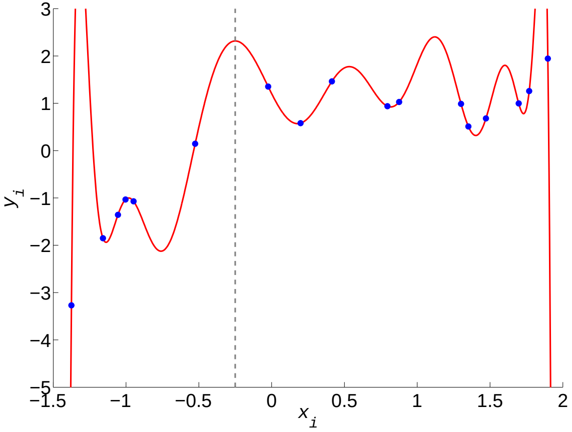
- The vector of training targets $\mathbf{y}$ lives in an N-dimensional vector space.
- The vector of training predictions $\mathbf{f}$ lives in the same space, but it is constrained to being generated by the $M+1$ columns of matrix $\boldsymbol{\Phi}$.
- The error vector $\mathbf{e}$ is minimal if it is orthogonal to all columns of $\boldsymbol{\Phi}$:

$$\boldsymbol{\Phi}^\top\,\mathbf{e} \;=\; 0 \;\;\Longleftrightarrow\;\; \boldsymbol{\Phi}^\top\,(\mathbf{y}-\boldsymbol{\Phi}\,\mathbf{w}) \;=\; 0$$

Least squares fit for polynomials of degree 0 to 17

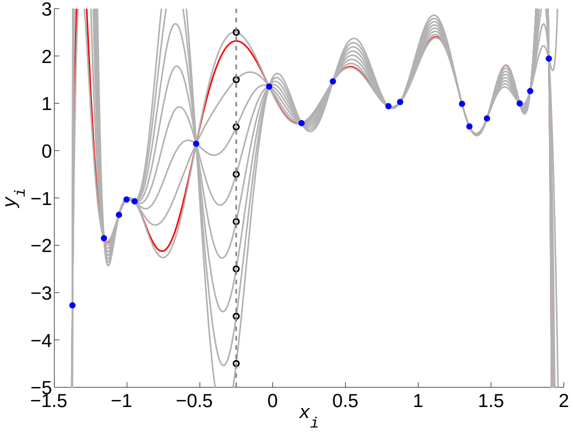


Have we solved the problem?



- Ok, so have we solved the problem?
- What do we think y_* is for $x_* = -0.25$? And for $x_* = 2$?
- If M is large enough, we can find a model that fits the data

Overfitting



- All the models in the figure are polynomials of degree 17 (18 weights).
- All perfectly fit the 17 training points, plus any desired y_* at $x_* = -0.25$.
- We have not solved the problem. Key missing ingredient: **assumptions!**

A laundry list of open questions

- Do we think that all models are equally probable... before we see any data?

  What does the probability of a model even mean?

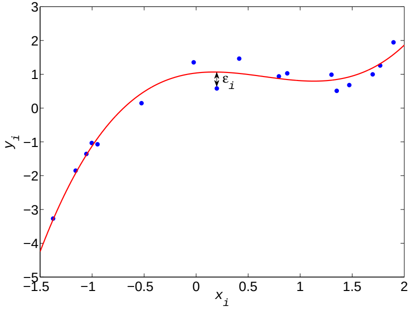
- Do we need to choose a single "best" model or can we consider several?

  We need a "language" to represent them.

- Perhaps our training targets are contaminated with noise. What to do?

  This question is a bit easier, we will start here.

Observation noise



- Imagine the data was in reality generated by the red function.
- But each $f(x_*)$ was independently contaminated by a noise term ϵ_i.
- The observations are noisy: $y_i = f_{\mathbf{w}}(x_i) + \epsilon_i$.
- We can characterise the noise with a probability density function. For example a Gaussian density function, $\epsilon_i \sim \mathcal{N}(\epsilon_i;\ 0, \sigma^2_{\text{noise}})$:

$$p(\epsilon_i) \;=\; \frac{1}{\sqrt{2\pi\,\sigma^2_{\text{noise}}}} \exp\Big(- \frac{\epsilon_i^2}{2\,\sigma^2_{\text{noise}}} \Big)$$

Probability of the observed data given the model

A vector and matrix notation view of the noise.

- $\boldsymbol{\epsilon} = [\epsilon_1, \ldots, \epsilon_N]^\top$ stacks the independent noise terms:

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon};\ \mathbf{0},\ \sigma^2_{\text{noise}}\mathbf{I}) \qquad p(\boldsymbol{\epsilon}) = \prod_{i=1}^{N} p(\epsilon_i) = \left(\frac{1}{\sqrt{2\pi\,\sigma^2_{\text{noise}}}}\right)^N \exp\left(-\frac{\boldsymbol{\epsilon}^\top\boldsymbol{\epsilon}}{2\,\sigma^2_{\text{noise}}}\right)$$

- Given that $\mathbf{y} = \mathbf{f} + \boldsymbol{\epsilon}$ we can write the probability of $\mathbf{y}$ given $\mathbf{f}$:

$$p(\mathbf{y}|\mathbf{f},\ \sigma^2_{\text{noise}}) = \mathcal{N}(\mathbf{y};\ \mathbf{f},\ \sigma^2_{\text{noise}}) = \left(\frac{1}{\sqrt{2\pi\,\sigma^2_{\text{noise}}}}\right)^N \exp\left(-\frac{\|\mathbf{y}-\mathbf{f}\|^2}{2\,\sigma^2_{\text{noise}}}\right)$$

$$= \left(\frac{1}{\sqrt{2\pi\,\sigma^2_{\text{noise}}}}\right)^N \exp\left(-\frac{E(\mathbf{w})}{2\,\sigma^2_{\text{noise}}}\right)$$

- $E(\mathbf{w}) = \sum_{i=1}^{N}(y_i - f_{\mathbf{w}}(x_i))^2 = \|\mathbf{y} - \boldsymbol{\Phi}\,\mathbf{w}\|^2$ is the sum of squared errors.
- Since $\mathbf{f} = \boldsymbol{\Phi}\,\mathbf{w}$ we can write $p(\mathbf{y}|\mathbf{w},\ \sigma^2_{\text{noise}}) = p(\mathbf{y}|\mathbf{f},\ \sigma^2_{\text{noise}})$ for a given $\boldsymbol{\Phi}$.

Likelihood function

Likelihood of the weights and probability of the data.

- $p(\mathbf{y}|\mathbf{w}, \sigma^2_{\text{noise}})$ is the probability of the observed data given the weights.
- $\mathcal{L}(\mathbf{w}) \propto p(\mathbf{y}|\mathbf{w}, \sigma^2_{\text{noise}})$ is the likelihood of the weights given the observed data.

Maximum likelihood.

- We can fit the model weights to the data by maximising the likelihood:

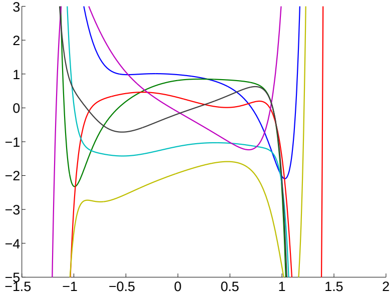
$$\hat{\mathbf{w}} = \operatorname{argmax} \mathcal{L}(\mathbf{w}) = \operatorname{argmax} \exp\left(-\frac{E(\mathbf{w})}{2\,\sigma^2_{\text{noise}}}\right) = \operatorname{argmin} E(\mathbf{w})$$

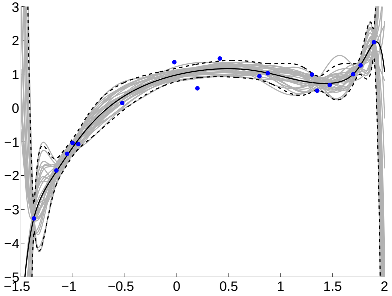
- With an additive Gaussian independent noise model, the maximum likelihood and the least squares solutions are the same.
- So ... we still have not solved the prediction problem! We still overfit.

Multiple explanations of the data

Multiple explanations:

- We do not know what particular function generated the data.
- More than one of our models can perfectly fit the data.
- We want to reason in terms of a set of possible explanations, not just one.
- We believe more than one of our models could have generated the data.





Model complexity:

- We do not believe all models are equally likely to explain the data.
- We may believe a simpler model is more likely than a complex one.

Medical inference (diagnosis)

Breast cancer facts:

- 1% of scanned women have breast cancer
- 80% of women with breast cancer get positive mammography
- 9.6% of women without breast cancer also get positive mammography

Question: A woman get's a scan, and it is positive; what is the probability that she has breast cancer?

1. less than 1%
2. around 10%
3. around 90%
4. more than 99%

Medical inference, numerical

Define: C = presence of breast cancer, $\bar{C}$ = no cancer

The probability of cancer for scanned women is $p(C) = 1\%$

If there is cancer, the probability of a positive mammography is $p(M|C) = 80\%$

If there is no cancer, we still have $p(M|\bar{C}) = 9.6\%$

The question is what is $p(C|M)$?

Consider 10000 subjects of screening

- $p(C) = 1\%$, therefore 100 of them have cancer, of which
    - $p(M|C) = 80\%$, therefore 80 get a positive mammography
    - 20 get a negative mammography
- $p(\bar{C}) = 99\%$, therefore 9900 of them do not have cancer, of which
    - $p(M|\bar{C}) = 9.6\%$, therefore 950 get a positive mammography
    - 8950 get a a negative mammography

	M	$\bar{M}$
C	80	20
$\bar{C}$	950	8950

$p(C|M)$ is obtained as the proportion of all positive mammographies for which there actually is breast cancer

$$p(C|M) \;=\; \frac{p(C,M)}{p(C,M)+p(\bar{C},M)} \;=\; \frac{p(C,M)}{p(M)} \;=\; \frac{80}{80+950} \;\simeq\; 7.8\%$$

This is an example of Bayes' rule:

$$p(A|B) \;=\; \frac{p(B|A)p(A)}{p(B)}.$$

Which is just a consequence of the definition of *conditional probability*

$$p(A|B) \;=\; \frac{p(A,B)}{p(B)}, \quad \text{(where } p(B) \neq 0\text{).}$$

Just two rules of probability theory

Astonishingly, the rich theory of probability can be derived using just two rules:

The *sum rule* states that

$$p(A) = \sum_B p(A, B), \quad \text{or} \quad p(A) = \int_B p(A, B)dB,$$

for discrete and continuous variables. Sometimes called *marginalization*.

The *product rule* states that

$$p(A, B) = p(A|B)p(B).$$

It follows directly from the definition of conditional probability, and leads directly to Bayes' rule

$$p(A|B)p(B) = p(A, B) = p(B|A)p(A) \Rightarrow p(A|B) = \frac{p(B|A)p(A)}{p(B)}.$$

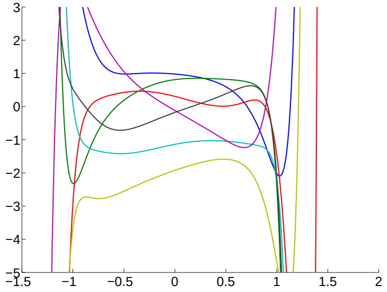
Special case:
if A and B are *independent*, $p(A|B) = p(A)$, and thus $p(A, B) = p(A)p(B)$.

Posterior probability of a function

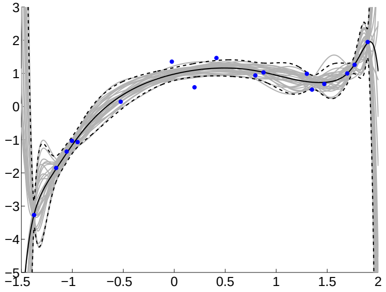
Given the prior functions $p(\mathbf{f})$ how can we make predictions?

- Of all functions generated from the prior, keep those that fit the data.
- The notion of closeness to the data is given by the likelihood $p(\mathbf{y}|\mathbf{f})$.
- We are really interested in the posterior distribution over functions:

$$p(\mathbf{f}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{f})\,p(\mathbf{f})}{p(\mathbf{y})} \qquad \text{Bayes Rule}$$



Some samples from the prior



Samples from the posterior

Prior probability of a function

A model $\mathcal{M}$ is the choice of a model structure and of parameter values.

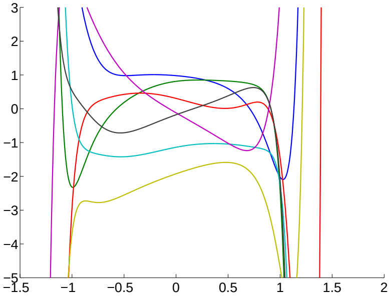
$$f_{\mathbf{w}}(x) \;=\; \sum_{j=0}^{M} w_j \, \phi_j(x)$$

The prior $p(\mathbf{w}|\mathcal{M})$ determines what functions this model can generate. Example:

- Imagine we choose $M = 17$, and $p(w_i) = \mathcal{N}(w_i;\ 0, \sigma_{\mathbf{w}}^2)$.
- We have actually defined a prior distribution over functions $p(\mathbf{f}|\mathcal{M})$.

This figure is generated as follows:

- Use polynomial basis functions, $\phi_j(x) = x^j$.
- Define a uniform grid of x values in [-1.5, 2].
- Generate matrix $\boldsymbol{\Phi}$ for $M = 17$.
- Set $\sigma_{\mathbf{w}}^2 = 1$ and sample w_i values.
- Compute $\mathbf{f} = \boldsymbol{\Phi}\, \mathbf{w}$ and plot it.



Maximum likelihood, parametric model

Supervised parametric learning:

- data: $\mathbf{x}, \mathbf{y}$
- model $\mathcal{M}$: $y = f_{\mathbf{w}}(x) + \varepsilon$

Gaussian likelihood:

$$p(\mathbf{y}|\mathbf{x}, \mathbf{w}, \mathcal{M}) \;\propto\; \prod_{i=1}^{N} \exp(-\tfrac{1}{2}(y_i - f_{\mathbf{w}}(x_i))^2/\sigma^2_{\text{noise}}).$$

Maximize the likelihood:

$$\mathbf{w}_{\text{ML}} \;=\; \operatorname*{argmax}_{\mathbf{w}} p(\mathbf{y}|\mathbf{x}, \mathbf{w}, \mathcal{M}).$$

Make predictions, by plugging in the ML estimate:

$$p(y_*|x_*, \mathbf{w}_{\text{ML}}, \mathcal{M})$$

Bayesian Inference, parametric model

Supervised parametric learning:

- data: $\mathbf{x}, \mathbf{y}$
- model $\mathcal{M}$: $y = f_{\mathbf{w}}(x) + \varepsilon$

Gaussian likelihood:

$$p(\mathbf{y}|\mathbf{x}, \mathbf{w}, \mathcal{M}) \;\propto\; \prod_{i=1}^{N} \exp(-\tfrac{1}{2}(y_i - f_{\mathbf{w}}(x_i))^2/\sigma^2_{\text{noise}}).$$

Parameter prior:

$$p(\mathbf{w}|\mathcal{M})$$

Posterior parameter distribution by Bayes rule $p(a|b) = p(b|a)p(a)/p(b)$:

$$p(\mathbf{w}|\mathbf{x}, \mathbf{y}, \mathcal{M}) \;=\; \frac{p(\mathbf{w}|\mathcal{M})p(\mathbf{y}|\mathbf{x}, \mathbf{w}, \mathcal{M})}{p(\mathbf{y}|\mathbf{x}, \mathcal{M})}$$

Bayesian inference, parametric model, cont.

Making predictions (marginalizing out the parameters):

$$
\begin{aligned}
p(y_*|x_*, \mathbf{x}, \mathbf{y}, \mathcal{M}) &= \int p(y_*, \mathbf{w}|\mathbf{x}, \mathbf{y}, x_*, \mathcal{M}) d\mathbf{w} \\
&= \int p(y_*|\mathbf{w}, x_*, \mathcal{M}) p(\mathbf{w}|\mathbf{x}, \mathbf{y}, \mathcal{M}) d\mathbf{w}.
\end{aligned}
$$

Marginal likelihood:

$$
p(\mathbf{y}|\mathbf{x}, \mathcal{M}) = \int p(\mathbf{w}|\mathcal{M}) p(\mathbf{y}|\mathbf{x}, \mathbf{w}, \mathcal{M}) d\mathbf{w}.
$$

Second level inference, model comparison, Bayes' rule again

$$
p(\mathcal{M}|\mathbf{y}, \mathbf{x}) = \frac{p(\mathbf{y}|\mathbf{x}, \mathcal{M}) p(\mathcal{M})}{p(\mathbf{y}|\mathbf{x})} \propto p(\mathbf{y}|\mathbf{x}, \mathcal{M}) p(\mathcal{M}).
$$

The *marginal likelihood* is used to select between models.

Some useful Gaussian identities

If x is multivariate Gaussian with mean μ and covariance matrix Σ

$$p(\mathbf{x}; \mu, \Sigma) \;=\; (2\pi|\Sigma|)^{-D/2} \exp\big(-(\mathbf{x}-\mu)^\top \Sigma^{-1} (\mathbf{x}-\mu)/2\big),$$

then

$$\begin{aligned} \mathbb{E}[\mathbf{x}] &= \mu, \\ \mathbb{V}[\mathbf{x}] &= \mathbb{E}[(\mathbf{x}-\mathbb{E}[\mathbf{x}])^2] \;=\; \Sigma. \end{aligned}$$

For any matrix A, if $\mathbf{z} = A\mathbf{x}$ then

$$\begin{aligned} \mathbb{E}[\mathbf{z}] &= A\mu, \\ \mathbb{V}[\mathbf{z}] &= A\Sigma A^\top. \end{aligned}$$

Marginal likelihood and predictive distribution in detail

Marginal likelihood for a linear in the parameters model with i.i.d. Gaussian noise:

- Gaussian *prior* on the weights: $p(\mathbf{w}|\mathcal{M}) = \mathcal{N}(\mathbf{w};\ \mathbf{0},\ \sigma_{\mathbf{w}}^2\,\mathbf{I})$
- Gaussian *likelihood* of the weights: $p(\mathbf{y}|\mathbf{x},\mathbf{w},\mathcal{M}) = \mathcal{N}(\mathbf{y};\ \mathbf{\Phi}\,\mathbf{w},\ \sigma_{\text{noise}}^2\,\mathbf{I})$

$$p(\mathbf{y}|\mathbf{x},\mathcal{M}) \;=\; \int p(\mathbf{w}|\mathcal{M})p(\mathbf{y}|\mathbf{x},\mathbf{w},\mathcal{M})d\mathbf{w} \;=\; \mathcal{N}(\mathbf{y};\ \mathbf{0}, \sigma_{\mathbf{w}}^2\,\mathbf{\Phi}\,\mathbf{\Phi}^\top + \sigma_{\text{noise}}^2\,\mathbf{I})$$

Posterior parameter distribution by Bayes rule $p(a|b) = p(b|a)p(a)/p(b)$:

$$p(\mathbf{w}|\mathbf{x},\mathbf{y},\mathcal{M}) \;=\; \frac{p(\mathbf{w}|\mathcal{M})p(\mathbf{y}|\mathbf{x},\mathbf{w},\mathcal{M})}{p(\mathbf{y}|\mathbf{x},\mathcal{M})} \;=\; \mathcal{N}(\mathbf{w};\ \boldsymbol{\mu},\ \mathbf{\Sigma})$$

$$\mathbf{\Sigma} \;=\; \left(\sigma_{\text{noise}}^{-2}\mathbf{\Phi}^\top\mathbf{\Phi} + \sigma_{\mathbf{w}}^{-2}\,\mathbf{I}\right)^{-1} \quad \text{and} \quad \boldsymbol{\mu} \;=\; \left(\mathbf{\Phi}^\top\mathbf{\Phi} + \frac{\sigma_{\text{noise}}^2}{\sigma_{\mathbf{w}}^2}\,\mathbf{I}\right)^{-1}\mathbf{\Phi}^\top\mathbf{y}$$

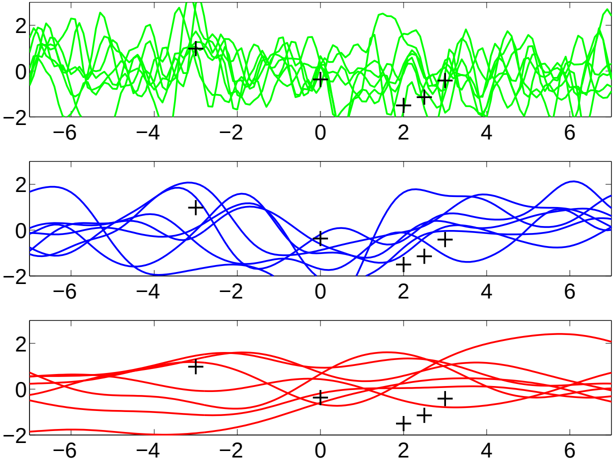
The predictive distribution is given by:

$$p(y_*|x_*,\mathbf{x},\mathbf{y},\mathcal{M}) \;=\; \mathcal{N}(y_*;\ \boldsymbol{\phi}(x_*)^\top\boldsymbol{\mu},\ \boldsymbol{\phi}(x_*)^\top\mathbf{\Sigma}\boldsymbol{\phi}(x_*) + \sigma_{\text{noise}}^2\,\mathbf{I}).$$

Understanding the marginal likelihood (1). Models

Consider 3 models $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$. Given our data:

- We want to compute the *marginal likelihood* for each model.
- We want to obtain the predictive distribution for each model.



Understanding the marginal likelihood (2). Noise

Consider a very simple noise model

- $\epsilon_i \sim \text{Uniform}(-0.2, 0.2)$ and all noise terms are independent.
- $p(y_i|f(x_i)) = 0$ if $|y_i - f(x_i)| > 0.2$, and $p(y_i|f(x_i)) = 1/0.4 = 2.5$ otherwise.
- The likelihood of a given function from the prior is

$$p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^{N} p(y_i|f(x_i)) = \begin{cases} 0 & \text{if any } |y_i - f(x_i)| > 0.2 \\ 2.5^N & \text{otherwise} \end{cases}$$

We will approximate the marginal likelihood by *rejection sampling*:

$$p(\mathbf{y}|\mathcal{M}_i) = \int p(\mathbf{y}|\mathbf{f})\, p(\mathbf{f}|\mathcal{M}_i)\, d\,\mathbf{f} \approx \frac{1}{S}\sum_{s=1}^{S} p(\mathbf{y}|\mathbf{f}_s) = \frac{S_a}{S} \cdot 2.5^N$$

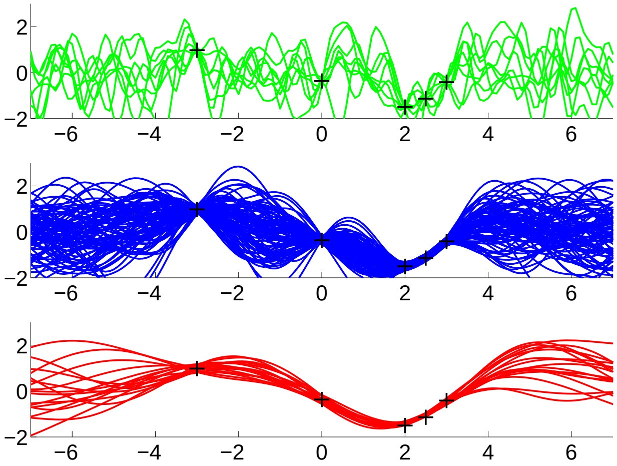
- $\mathbf{f}_s$ is a sample function drawn from the prior $p(\mathbf{f}|\mathcal{M}_i)$.
- A total of S functions are sampled from the prior.
- S_a is the number of samples with non-zero likelihood: these are accepted. The remaining $S - S_a$ samples are rejected.

Understanding the marginal likelihood (3). Posterior

Posterior samples for each of the models obtained by rejection sampling.

- For each model we draw 1 million samples from the prior.
- We only keep the samples that have non-zero likelihood.

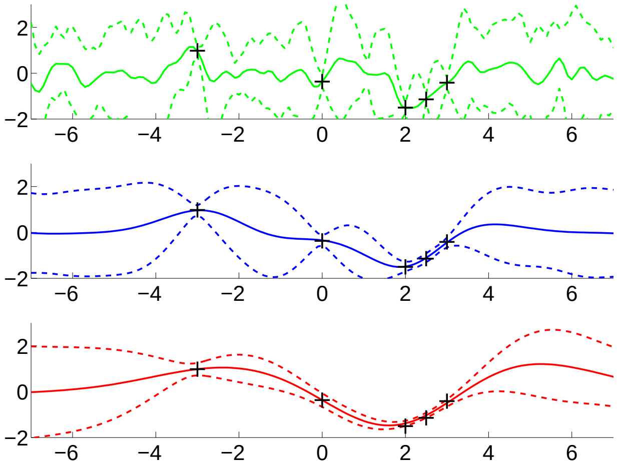
S_a	$p(\mathbf{y}\mid\mathcal{M}_i)$
8	8.10^{-4}
88	$\mathbf{9.10^{-3}}$
17	2.10^{-3}



Predictive distribution

Predictive distribution for each of the models obtained.

- For each model we take all the posterior functions from rejection sampling.
- We compute the average and standard deviation of $f_s(x_i)$.



Conclusions

Probability theory provides a framework for

- making inference in a model
- making probabilistic predictions

in parametric models.

It also provides a *principled* and *automatic* way of doing

- model comparison

In the follwing lectures, we'll demonstrate how to use this framework to solve challenging machine learning problems.