Announcements

Matlab Grader homework, emailed Thursday,
1 (of 9) homeworks Due 21 April, Binary graded.
2 this week

Jupyter homework?: translate matlab to Jupiter, TA Harshul h6gupta@eng.ucsd.edu or me
I would like this to happen.

“GPU” homework. NOAA climate data in Jupyter on the datahub.ucsd.edu, 15 April.

Projects: Any computer language

Podcast might work eventually.

Today:

- Stanford CNN
- Gaussian, Bishop 2.3
- Gaussian Process 6.4
- Linear regression 3.0-3.2

Wednesday 10 April
Stanford CNN, Linear models for regression 3, Applications of Gaussian processes.

Bayes and Softmax (Bishop p. 198)

- Bayes:

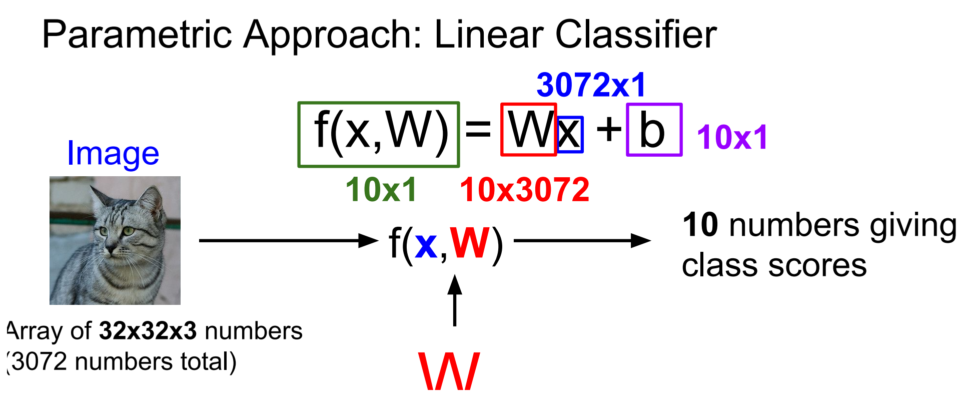
$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{p(y|x)p(x)}{\sum_{y \in Y} p(x,y)}$$

- Classification of N classes:

$$p(\mathcal{C}_n|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_n)p(\mathcal{C}_n)}{\sum_{k=1}^{N} p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)} = \frac{\exp(a_n)}{\sum_{k=1}^{N} \exp(a_k)}$$

with

$$a_n = \ln\left(p(\mathbf{x}|\mathcal{C}_n)p(\mathcal{C}_n)\right)$$


Parametric Approach: Linear Classifier
3072x1
f(x,W) = Wx + b
10x1
10x1
10x3072
Image
f(x,W)
10 numbers giving class scores
Array of 32x32x3 numbers (3072 numbers total)
W

Softmax to Logistic Regression (Bishop p. 198)

$$
\begin{aligned}
p(\mathcal{C}_1|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{\sum_{k=1}^{2} p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)} \\
&= \frac{\exp(a_1)}{\sum_{k=1}^{2} \exp(a_k)} = \frac{1}{1+\exp(-a)}
\end{aligned}
$$

with

$$
a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}
$$

Softmax with Gaussian(Bishop p. 198)

$$p(\mathcal{C}_n|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_n)p(\mathcal{C}_n)}{\sum_{k=1}^{N} p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}$$
$$= \frac{\exp(a_n)}{\sum_{k=1}^{N} \exp(a_k)}$$

with

$$a_n = \ln\left(p(\mathbf{x}|\mathcal{C}_n)p(\mathcal{C}_n)\right)$$

Assuming $\mathbf{x}$ is Gaussian $\mathcal{N}(\mu_n, \mathbf{\Sigma})$

[illegible]

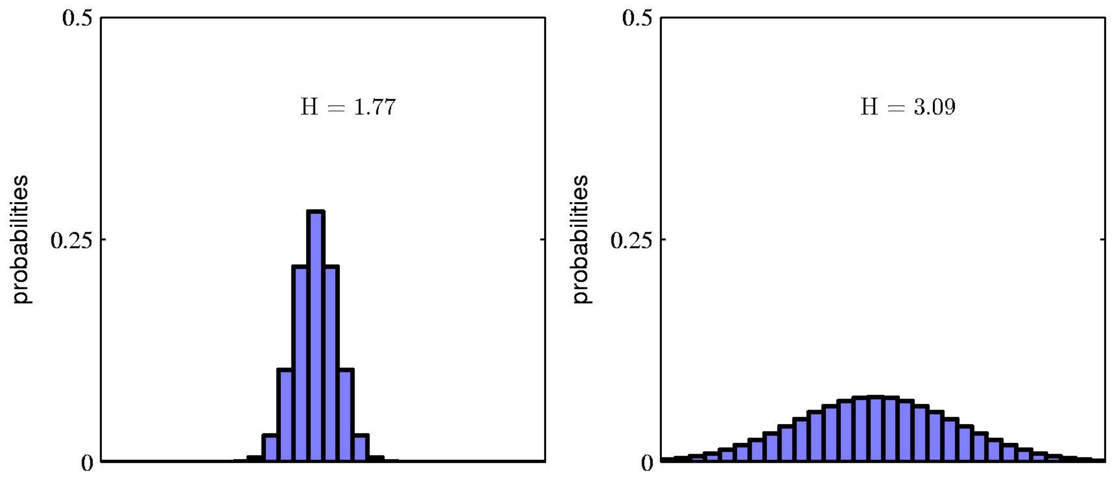
$$a_n = \mathbf{w}_n^T\mathbf{x} + w_0$$
$$\mathbf{w}_n = \mathbf{\Sigma}^{-1}\mu_n$$
$$w_0 = \frac{-1}{2}\mu_n^T\mathbf{\Sigma}^{-1}\mu_n + \ln(p(\mathcal{C}_n))$$

Entropy 1.6

$$\mathrm{H}[x] = -\sum_{x} p(x) \log_2 p(x)$$

Important quantity in

- coding theory
- statistical physics
- machine learning



The Kullback-Leibler Divergence

P true distribution, q is approximating distribution

$$\begin{aligned}\mathrm{KL}(p\|q) &= -\int p(\mathbf{x}) \ln q(\mathbf{x})\,\mathrm{d}\mathbf{x} - \left(-\int p(\mathbf{x}) \ln p(\mathbf{x})\,\mathrm{d}\mathbf{x}\right) \\ &= -\int p(\mathbf{x}) \ln \left\{\frac{q(\mathbf{x})}{p(\mathbf{x})}\right\} \mathrm{d}\mathbf{x}\end{aligned}$$

$$\mathrm{KL}(p\|q) \geqslant 0 \qquad\qquad \mathrm{KL}(p\|q) \not\equiv \mathrm{KL}(q\|p)$$

KL homework

- Support of P and Q = >   “only >0” don’t use isnan isinf

- After you pass. Take your time to clean up. Get close to 50

Lecture 3

- Homework

- Pod-cast lecture on-line

- Next lectures:
  - I posted a rough plan.
  - It is flexible though so please come with suggestions

Bayes for linear model

$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{n}$ $\quad$ $\boldsymbol{n} \sim \mathrm{N}(\boldsymbol{0}, \boldsymbol{C}_n)$ $\quad$ $\mathbf{y} \sim \mathrm{N}(\boldsymbol{A}\boldsymbol{x}, \boldsymbol{C}_n)$ $\quad$ prior: $\boldsymbol{x} \sim \mathrm{N}(\boldsymbol{0}, \boldsymbol{C}_x)$

$p(\boldsymbol{x}|\boldsymbol{y}) \sim p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x}) \sim N\left(\boldsymbol{x}_p, \boldsymbol{C}_p\right)$

mean $\quad \boldsymbol{x}_p = \boldsymbol{C}_p \boldsymbol{A}^T \boldsymbol{C}_n^{-1} \boldsymbol{y}$

Covariance $\quad \boldsymbol{C}_p^{-1} = \boldsymbol{A}^T \boldsymbol{C}_n^{-1} \boldsymbol{A} + \boldsymbol{C}_x^{-1}$

Bayes' Theorem for Gaussian Variables

- Given

$$\begin{aligned} p(\mathbf{x}) &= \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}\right) \\ p(\mathbf{y}|\mathbf{x}) &= \mathcal{N}\left(\mathbf{y}|\mathbf{A}\mathbf{x}+\mathbf{b}, \mathbf{L}^{-1}\right) \end{aligned}$$

- we have

$$\begin{aligned} p(\mathbf{y}) &= \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu}+\mathbf{b}, \mathbf{L}^{-1}+\mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}}) \\ p(\mathbf{x}|\mathbf{y}) &= \mathcal{N}(\mathbf{x}|\boldsymbol{\Sigma}\{\mathbf{A}^{\mathrm{T}}\mathbf{L}(\mathbf{y}-\mathbf{b})+\boldsymbol{\Lambda}\boldsymbol{\mu}\}, \boldsymbol{\Sigma}) \end{aligned}$$

- where

$$\boldsymbol{\Sigma} = (\boldsymbol{\Lambda}+\mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A})^{-1}$$

Sequential Estimation of mean (Bishop 2.3.5)

Contribution of the Nth data point, x_N

$$
\begin{aligned}
\boldsymbol{\mu}_{\mathrm{ML}}^{(N)} &= \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n \\
&= \frac{1}{N}\mathbf{x}_N + \frac{1}{N}\sum_{n=1}^{N-1}\mathbf{x}_n \\
&= \frac{1}{N}\mathbf{x}_N + \frac{N-1}{N}\boldsymbol{\mu}_{\mathrm{ML}}^{(N-1)} \\
&= \boldsymbol{\mu}_{\mathrm{ML}}^{(N-1)} + \frac{1}{N}\underbrace{(\mathbf{x}_N - \boldsymbol{\mu}_{\mathrm{ML}}^{(N-1)})}
\end{aligned}
$$

correction given x_N

correction weight

old estimate

Bayesian Inference for the Gaussian (Bishop2.3.6)

Assume σ^2 is known. Given i.i.d. data $\mathbf{x} = \{x_1, \ldots, x_N\}$ the likelihood function for μ is given by

$$p(\mathbf{x}|\mu) = \prod_{n=1}^{N} p(x_n|\mu) = \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left\{ -\frac{1}{2\sigma^2} \sum_{n=1}^{N} (x_n - \mu)^2 \right\}.$$

- This has a Gaussian shape as a function of μ (but it is *not* a distribution over μ).

Bayesian Inference for the Gaussian (Bishop2.3.6)

- Combined with a **Gaussian prior over μ,** $p(\mu) = \mathcal{N}\left(\mu|\mu_0, \sigma_0^2\right).$
- this gives the posterior

$$p(\mu|\mathbf{x}) \propto p(\mathbf{x}|\mu)p(\mu).$$

$$p(\mu|\mathbf{x}) = \mathcal{N}\left(\mu|\mu_N, \sigma_N^2\right)$$

$$\begin{aligned} \mu_N &= \frac{\sigma^2}{N\sigma_0^2 + \sigma^2}\mu_0 + \frac{N\sigma_0^2}{N\sigma_0^2 + \sigma^2}\mu_{\mathrm{ML}}, \\ \frac{1}{\sigma_N^2} &= \frac{1}{\sigma_0^2} + \frac{N}{\sigma^2}. \end{aligned}$$

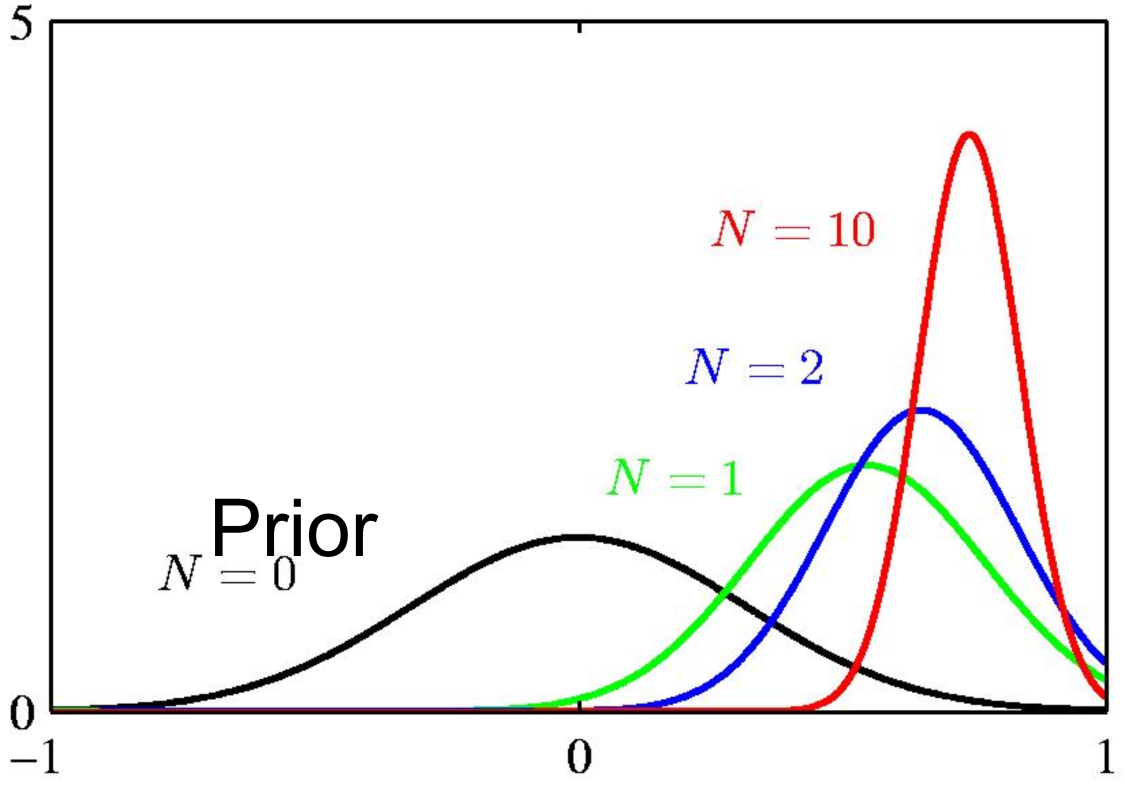
$$\mu_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N} x_n$$

	$N = 0$	$N \rightarrow \infty$
μ_N	μ_0	μ_{ML}
σ_N^2	σ_0^2	0

Bayesian Inference for the Gaussian (3)

- Example: for N = 0, 1, 2 and 10.

$$p(\mu|\mathbf{x}) = \mathcal{N}\left(\mu|\mu_N, \sigma_N^2\right)$$



Bayesian Inference for the Gaussian (4)

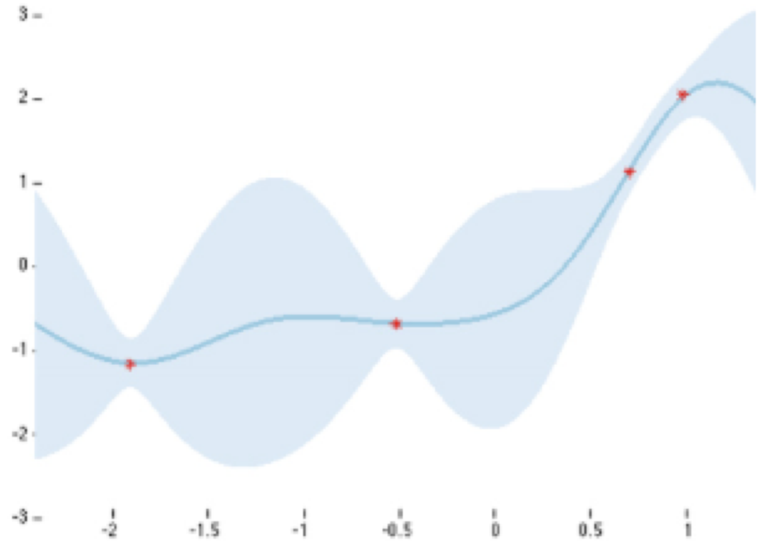
Sequential Estimation

$$\begin{aligned} p(\mu|\mathbf{x}) &\propto p(\mu)p(\mathbf{x}|\mu) \\ &= \left[p(\mu) \prod_{n=1}^{N-1} p(x_n|\mu) \right] p(x_N|\mu) \\ &\propto \mathcal{N}\left(\mu|\mu_{N-1}, \sigma_{N-1}^2\right) p(x_N|\mu) \end{aligned}$$

The posterior obtained after observing N-1 data points becomes the prior when we observe the Nth data point.

Conjugate prior: posterior and prior are in the same family. The **prior** is called a **conjugate prior** for the likelihood function.

Gaussian Process (Bishop 6.4, Murphy15)



This is what a Gaussian process *posterior* looks like with 4 data points and a squared exponential covariance function. The bold blue line is the predictive mean, while the light blue shade is the predictive uncertainty (2 standard deviations). The model uncertainty is small near the data, and increases as we move away from the data points.

$$t_n = y_n + \epsilon_n$$

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}'))$$

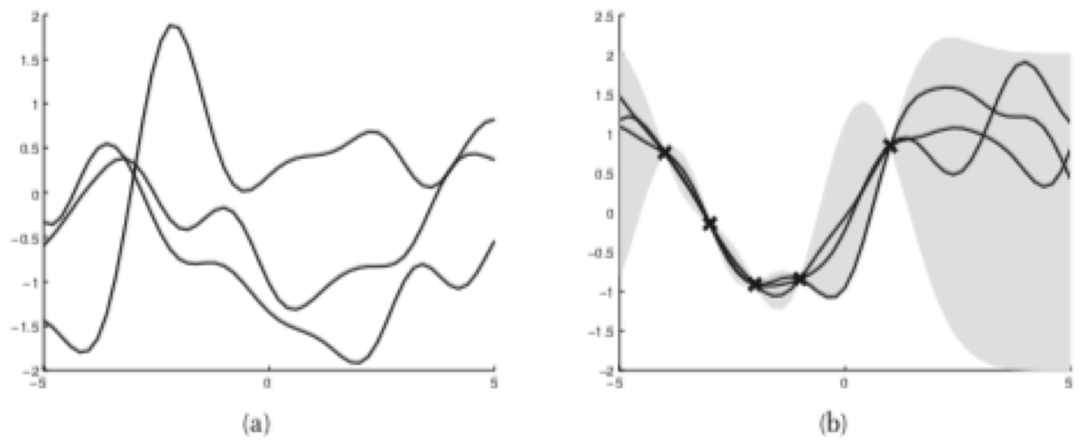


Figure 15.2 Left: some functions sampled from a GP prior with SE kernel. Right: some samples from a GP posterior, after conditioning on 5 noise-free observations. The shaded area represents $\mathbb{E}[f(\mathbf{x})] \pm 2\mathrm{std}(f(\mathbf{x}))$. Based on Figure 2.2 of (Rasmussen and Williams 2006). Figure generated by `gprDemoNoiseFree`.

Gaussian Process (Murphy ch15)

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}'))$$

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})] \\ \kappa(\mathbf{x}, \mathbf{x}') &= \mathbb{E}\left[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))^T\right] \end{aligned}$$

$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{f}|\boldsymbol{\mu}, \mathbf{K})$$

$K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ and $\boldsymbol{\mu} = (m(\mathbf{x}_1), \ldots, m(\mathbf{x}_N))$.

Training

$\mathcal{D} = \{(\mathbf{x}_i, f_i), i = 1 : N\}$, where $f_i = f(\mathbf{x}_i)$ is the noise-free

$$\begin{pmatrix} \mathbf{f} \\ \mathbf{f}_* \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu}_* \end{pmatrix}, \begin{pmatrix} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^T & \mathbf{K}_{**} \end{pmatrix}\right)$$

$\mathbf{K} = \kappa(\mathbf{X}, \mathbf{X})$ is $N \times N$, $\mathbf{K}_* = \kappa(\mathbf{X}, \mathbf{X}_*)$ is $N \times N_*$, and $\mathbf{K}_{**} = \kappa(\mathbf{X}_*, \mathbf{X}_*)$ is $N_* \times N_*$.

Gaussian Process (Murphy ch15)

$$\begin{pmatrix} \mathbf{f} \\ \mathbf{f}_* \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu}_* \end{pmatrix}, \begin{pmatrix} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^T & \mathbf{K}_{**} \end{pmatrix} \right)$$

The conditional is Gaussian:

$$\begin{aligned} p(\mathbf{f}_*|\mathbf{X}_*, \mathbf{X}, \mathbf{f}) &= \mathcal{N}(\mathbf{f}_*|\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) \\ \boldsymbol{\mu}_* &= \boldsymbol{\mu}(\mathbf{X}_*) + \mathbf{K}_*^T \mathbf{K}^{-1}(\mathbf{f} - \boldsymbol{\mu}(\mathbf{X})) \\ \boldsymbol{\Sigma}_* &= \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{K}_* \end{aligned}$$

Common kernel is the squared exponential, RBF, Gaussian kernel

$$\kappa(x, x') = \sigma_f^2 \exp\left(-\frac{1}{2\ell^2}(x - x')^2\right)$$

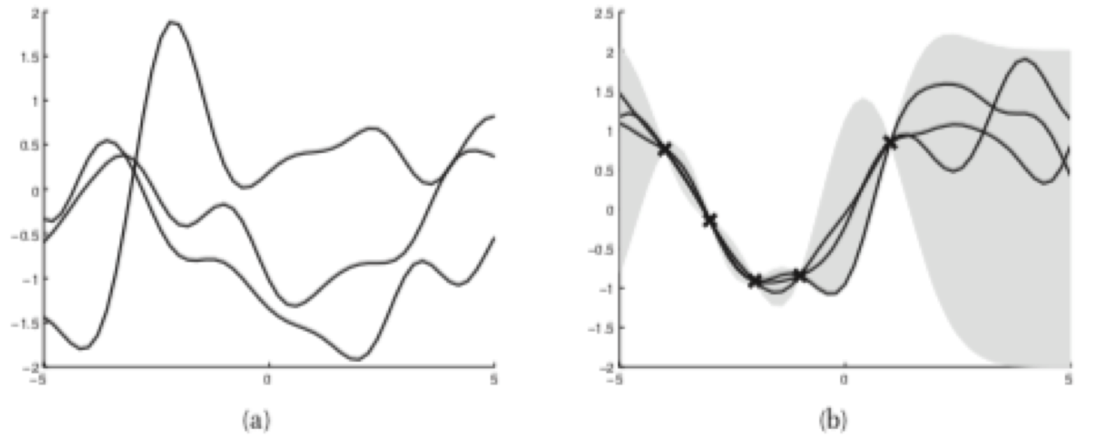


Figure 15.2 Left: some functions sampled from a GP prior with SE kernel. Right: some samples from a GP posterior, after conditioning on 5 noise-free observations. The shaded area represents $\mathbb{E}[f(\mathbf{x})] \pm 2\text{std}(f(\mathbf{x}))$. Based on Figure 2.2 of (Rasmussen and Williams 2006). Figure generated by `gprDemoNoiseFree`.

Gaussian Process (Bishop 6.4)

- Simple linear model $y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})$
- With prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$
- For multiple measurements

$$\mathbf{y} = \boldsymbol{\Phi}\mathbf{w}$$

$$\begin{aligned}\mathbb{E}[\mathbf{y}] &= \boldsymbol{\Phi}\mathbb{E}[\mathbf{w}] = \mathbf{0} \\ \operatorname{cov}[\mathbf{y}] &= \mathbb{E}\left[\mathbf{y}\mathbf{y}^{\mathrm{T}}\right] = \boldsymbol{\Phi}\mathbb{E}\left[\mathbf{w}\mathbf{w}^{\mathrm{T}}\right]\boldsymbol{\Phi}^{\mathrm{T}} = \frac{1}{\alpha}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} = \mathbf{K}\end{aligned}$$

where $\mathbf{K}$ is the Gram matrix with elements

$$K_{nm} = k(\mathbf{x}_n, \mathbf{x}_m) = \frac{1}{\alpha}\boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_m)$$

and $k(\mathbf{x}, \mathbf{x}')$ is the kernel function.

Gaussian Process (Bishop 6.4)

Measurement model

$$t_n = y_n + \epsilon_n$$

$$p(t_n|y_n) = \mathcal{N}(t_n|y_n, \beta^{-1})$$

Multiple Measurement model

$$p(\mathbf{t}|\mathbf{y}) = \mathcal{N}(\mathbf{t}|\mathbf{y}, \beta^{-1}\mathbf{I}_N)$$

Integrating out

$$p(\mathbf{t}) = \int p(\mathbf{t}|\mathbf{y})p(\mathbf{y})\,\mathrm{d}\mathbf{y} = \mathcal{N}(\mathbf{t}|\mathbf{0}, \mathbf{C})$$

$$k(\mathbf{x}_n, \mathbf{x}_m) = \theta_0 \exp\left\{-\frac{\theta_1}{2}\|\mathbf{x}_n - \mathbf{x}_m\|^2\right\}$$

$$C(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) + \beta^{-1}\delta_{nm}.$$

Predicting observation t_{N+1}

$$p(\mathbf{t}_{N+1}) = \mathcal{N}(\mathbf{t}_{N+1}|\mathbf{0}, \mathbf{C}_{N+1}) \quad \mathbf{C}_{N+1} = \begin{pmatrix} \mathbf{C}_N & \mathbf{k} \\ \mathbf{k}^{\mathrm{T}} & c \end{pmatrix}$$

The conditional $p(t_{N+1}|\, \mathbf{t}_{N+1})$ is Gaussian

$$\begin{aligned} m(\mathbf{x}_{N+1}) &= \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{t} \\ \sigma^2(\mathbf{x}_{N+1}) &= c - \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{k}. \end{aligned}$$

Nonparametric Methods (1) Bishop 2.5

- Parametric distribution models (... Gaussian) are restricted to specific forms, which may not always be suitable; for example, consider modelling a multimodal distribution with a single, unimodal model.

- Nonparametric approaches make few assumptions about the overall shape of the distribution being modelled.

- 1000 parameters versus 10 parameters

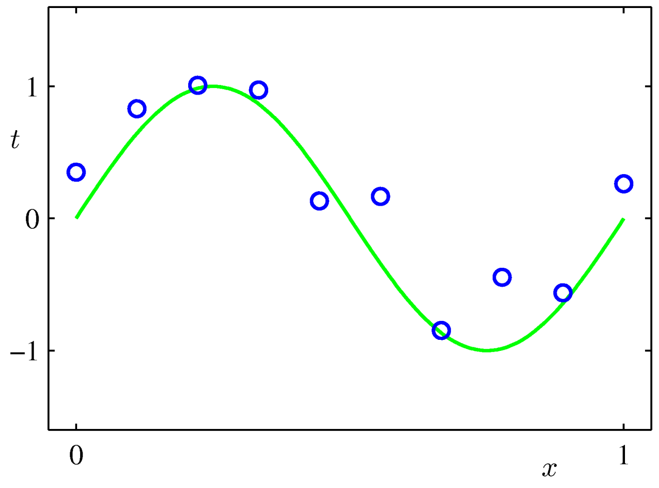
- Nonparametric models (not histograms) requires storing and computing with the entire data set.
- Parametric models, once fitted, are much more efficient in terms of storage and computation.

Linear regression: Linear Basis Function Models (1)

Generally

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x})$$

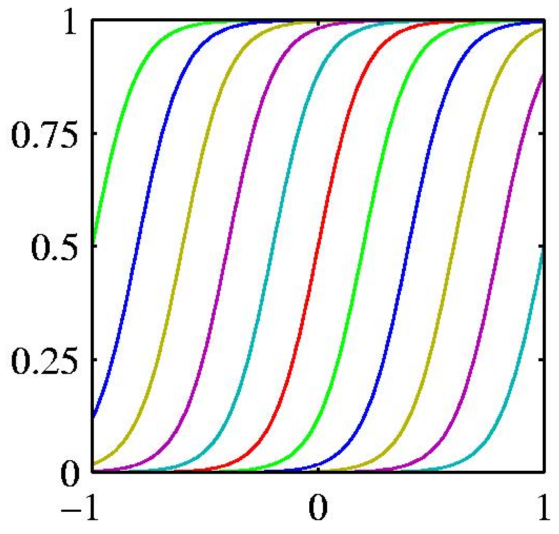
- where $\phi_j(x)$ are known as *basis functions*.
- Typically, $\phi_0(x) = 1$, so that w_0 acts as a bias.
- Simplest case is linear basis functions: $\phi_d(x) = x_d$.

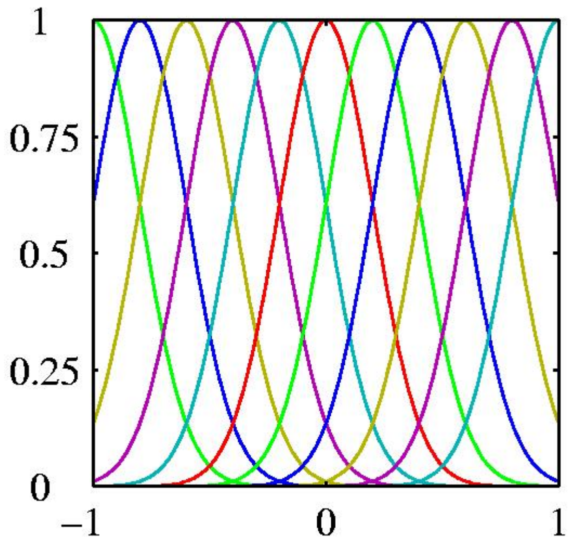


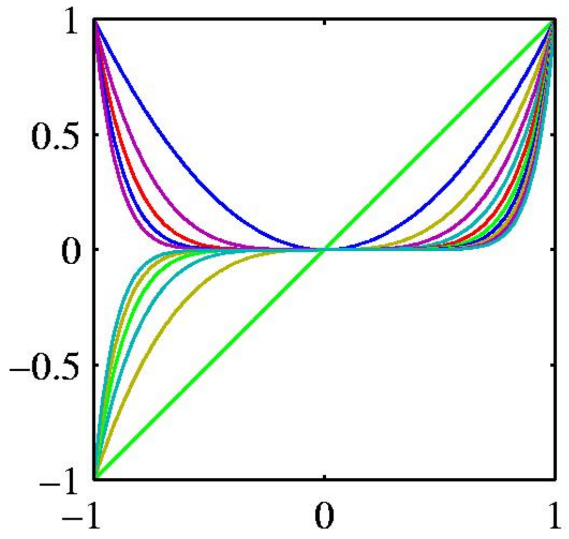
$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

http://playground.tensorflow.org/

Some types of basis function in 1-D







Sigmoids **Gaussians** **Polynomials**

$$\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right)$$

$$\sigma(a) = \frac{1}{1+\exp(-a)}.$$

$$\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$$

$$\phi_j(x) = x^j.$$

Sigmoid and Gaussian basis functions can also be used in multilayer neural networks, but neural networks learn the parameters of the basis functions. This is more powerful but also harder and messier.

Two types of linear model that are equivalent with respect to learning

$$y(\mathbf{x},\mathbf{w}) = \overset{\text{bias}}{\overset{\downarrow}{w_0}} + w_1 x_1 + w_2 x_2 + \ldots = \mathbf{w}^T \mathbf{x}$$

$$y(\mathbf{x},\mathbf{w}) = w_0 + w_1 \phi_1(\mathbf{x}) + w_2 \phi_2(\mathbf{x}) + \ldots = \mathbf{w}^T \Phi(\mathbf{x})$$

- The first and second model has the same number of adaptive coefficients as the number of basis functions +1.
- Once we have replaced the data by basis functions outputs, fitting the second model is exactly the same the first model.
  - No need to clutter math with basis functions

Maximum Likelihood and Least Squares (1)

- Assume observations from a deterministic function with added Gaussian noise:

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon \qquad \text{where} \qquad p(\epsilon|\beta) = \mathcal{N}(\epsilon|0, \beta^{-1})$$

- or,

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1}).$$

- Given observed inputs, $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, and targets $\mathbf{t} = [t_1, \ldots, t_N]^{\mathrm{T}}$, we obtain the likelihood function

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}).$$

Maximum Likelihood and Least Squares (2)

Taking the logarithm, we get

$$\begin{aligned}\ln p(\mathbf{t}|\mathbf{w},\beta) &= \sum_{n=1}^{N} \ln \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n),\beta^{-1}) \\ &= \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \beta E_D(\mathbf{w})\end{aligned}$$

Where the sum-of-squares error is

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

Maximum Likelihood and Least Squares (3)

Computing the gradient and setting it to zero yields

$$\nabla_{\mathbf{w}} \ln p(\mathbf{t}|\mathbf{w},\beta) = \beta \sum_{n=1}^{N} \left\{ t_n - \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n) \right\} \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}} = \mathbf{0}.$$

Solving for w,

$$\mathbf{w}_{\mathrm{ML}} = \underbrace{\left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}}\mathbf{t}$$

The Moore-Penrose pseudo-inverse, $\boldsymbol{\Phi}^{\dagger}$.

where

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}.$$

Maximum Likelihood and Least Squares (4)

Maximizing with respect to the bias, w_0, alone,

$$\begin{aligned} w_0 &= \bar{t} - \sum_{j=1}^{M-1} w_j \overline{\phi_j} \\ &= \overbrace{\frac{1}{N}\sum_{n=1}^{N} t_n}^{\bar{t}} - \sum_{j=1}^{M-1} w_j \overbrace{\frac{1}{N}\sum_{n=1}^{N} \phi_j(\mathbf{x}_n)}^{\overline{\phi_j}}. \end{aligned}$$

We can also maximize with respect to β, giving

$$\frac{1}{\beta_{\mathrm{ML}}} = \frac{1}{N}\sum_{n=1}^{N}\{t_n - \mathbf{w}_{\mathrm{ML}}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2$$

Geometry of Least Squares

Consider

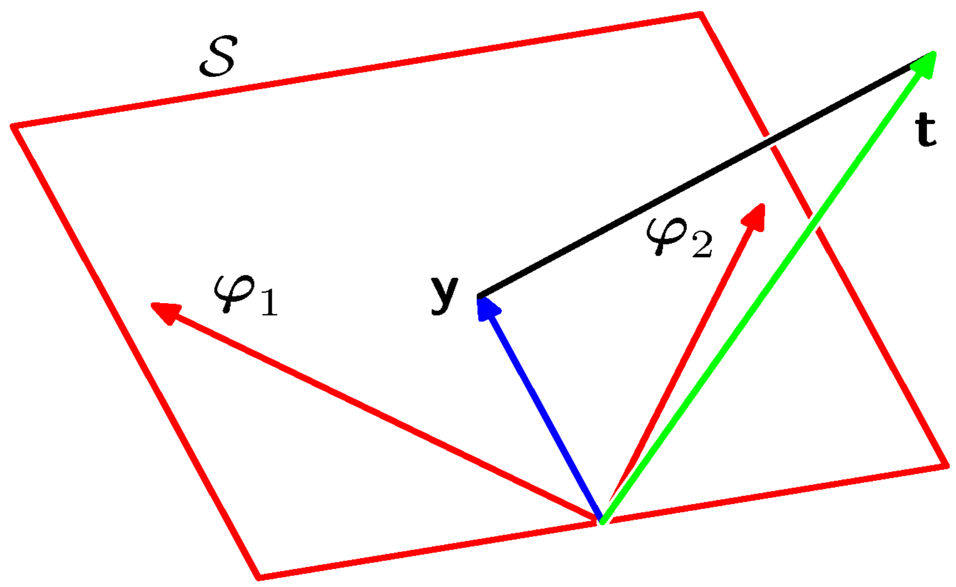
$$\mathbf{y} = \mathbf{\Phi}\mathbf{w}_{\mathrm{ML}} = [\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_M]\,\mathbf{w}_{\mathrm{ML}}.$$

$$\mathbf{y} \in \mathcal{S} \subseteq \mathcal{T} \qquad \mathbf{t} \in \mathcal{T}$$

N-dimensional

M-dimensional

S is spanned by $\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_M$



w$_{\mathrm{ML}}$ minimizes the distance between t and its orthogonal projection on S, i.e. y.

Least mean squares: An alternative approach for big datasets

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta \boxed{\nabla E_{n(\tau)}}$$

weights after seeing training case tau+1

learning rate

squared error derivatives w.r.t. the weights on the training case at time tau.

This is **"on-line" learning**. It is efficient if the dataset is redundant and simple to implement.

- It is called **stochastic gradient descent** if the training cases are picked randomly.
- Care must be taken with the learning rate to prevent divergent oscillations. Rate must decrease with tau to get a good fit.

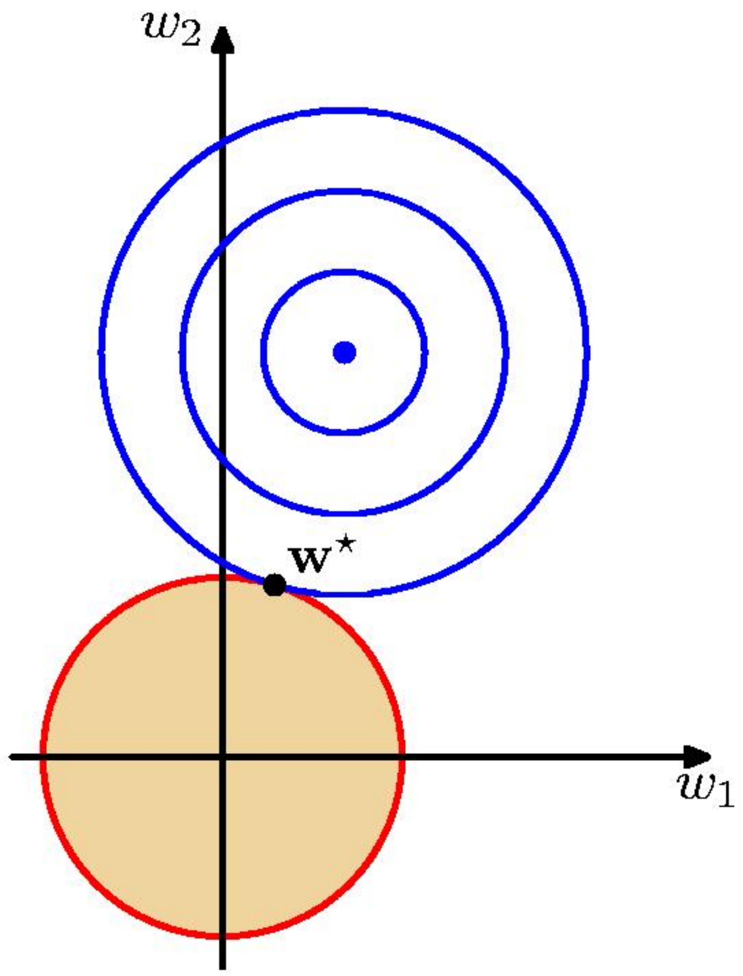
Regularized least squares

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(\mathbf{x}_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

The squared weights penalty is mathematically compatible with the squared error function, giving a closed form for the optimal weights:

$$\mathbf{w}^* = (\lambda \underset{\uparrow\ \text{identity matrix}}{\mathbf{I}} + \mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{t}$$

A picture of the effect of the regularizer



- The overall cost function is the sum of two parabolic bowls.
- The sum is also a parabolic bowl.
- The combined minimum lies on the line between the minimum of the squared error and the origin.
- The L2 regularizer just **shrinks** the weights.

A problem with the regularizer

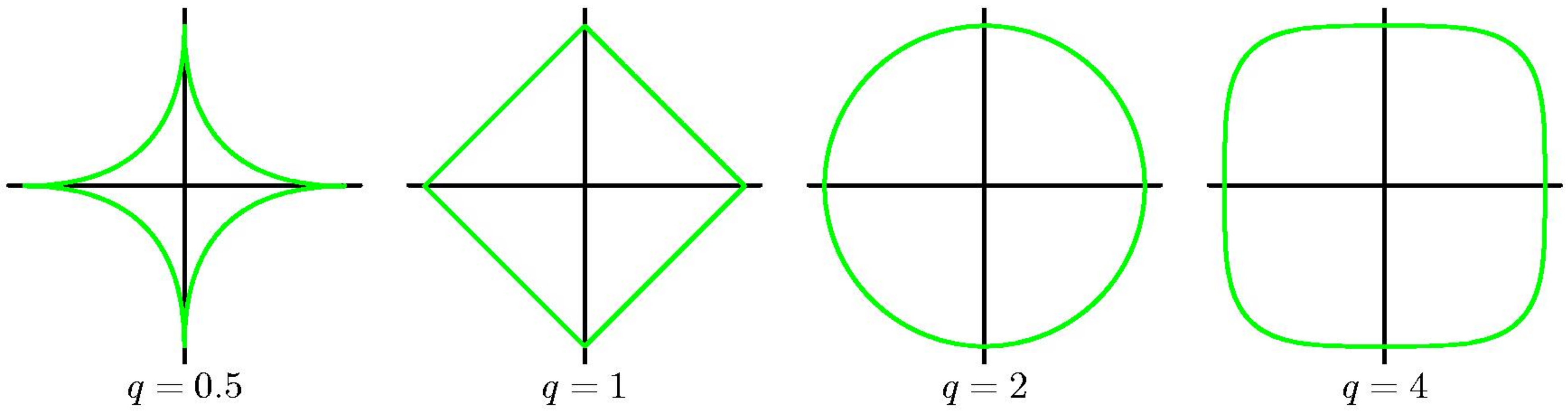
- The solution should be independent of the units we of the input vector.
- If components have different units (e.g. age and height), we have a problem.
  - If we measure age in months and height in meters, the relative values of the two weights are very different than if we use years and millimeters. The squared penalty has very different effects.
- A way to avoid the units problem: Whiten the data so that the input components all have unit variance and no covariance. This stops the regularizer from being applied to the whitening matrix.

$$\mathbf{X}^T_{whitened} = (\mathbf{X}^T \mathbf{X})^{-\frac{1}{2}} \mathbf{X}^T$$

  - … this can cause other problems when input components are almost perfectly correlated.
  - We really need a prior on the weight on each input component.

Other regularizers

- We do not need to use the squared error, provided we are willing to do more computation.
- Other powers of the weights can be used.

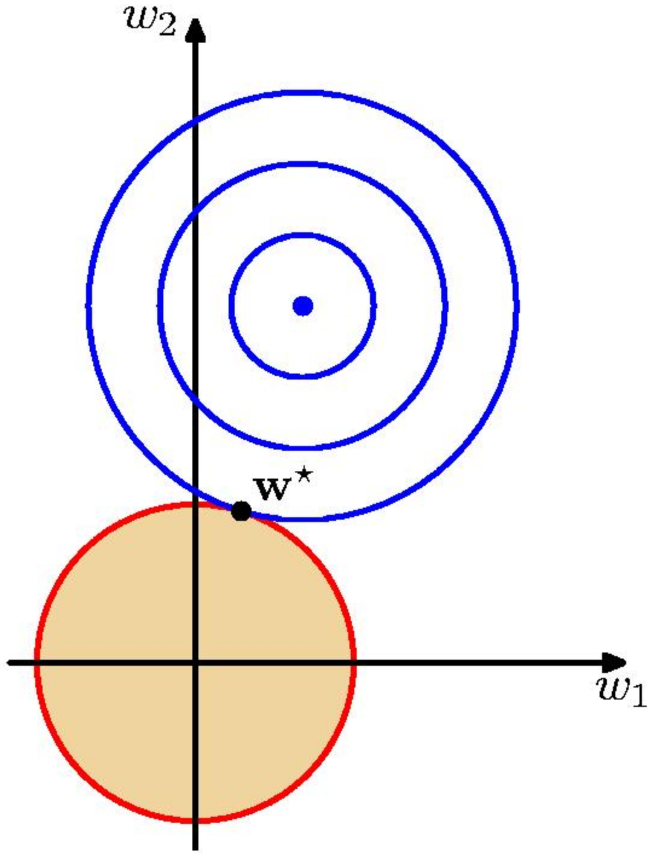


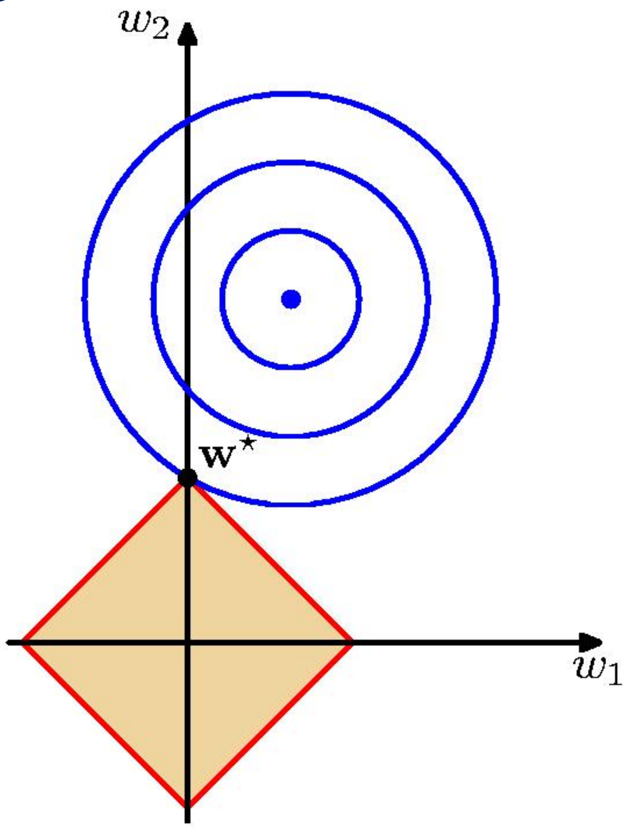
The lasso: penalizing the absolute values of the weights

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(\mathbf{x}_n, \mathbf{w}) - t_n\}^2 \quad + \quad \lambda \sum_i |\, \mathbf{w}_i \,|$$

- Finding the minimum requires quadratic programming but its still unique because the cost function is convex (a bowl plus an inverted pyramid)
- As lambda is increased, many of the weights go to exactly zero.
  - This is great for interpretation, and it is also pretty good for preventing overfitting.

Geometrical view of the lasso compared with a penalty on the squared weights





Notice **w1=0** at the optimum

Minimizing the absolute error

$$\min_{over\ \mathbf{w}} \quad \sum_n | t_n - \mathbf{w}^T \mathbf{x}_n |$$

- This minimization involves solving a linear programming problem.
- It corresponds to maximum likelihood estimation if the output noise is modeled by a Laplacian instead of a Gaussian.

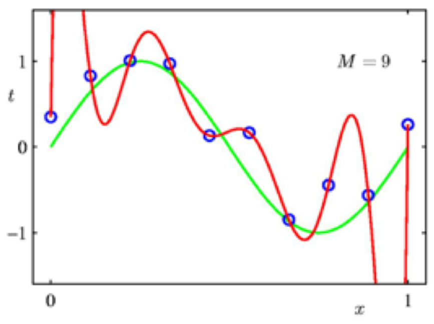
$$p(t_n \mid y_n) \ = \ a\, e^{-a\, |t_n - y_n|}$$

$$-\log p(t_n \mid y_n) \ = \ -a \,| t_n - y_n | + const$$

The bias-variance trade-off
(a figment of the frequentists lack of imagination?)

- Imagine a training set drawn at random from a whole set of training sets.
- The squared loss can be decomposed into a
  - Bias = systematic error in the model's estimates
  - Variance = noise in the estimates cause by sampling noise in the training set.
- There is also additional loss due to that the target values are noisy.
  - We eliminate this extra, irreducible loss from the math by using the average target values (i.e. the unknown, noise-free values)

9 Order Polynomial



The bias-variance decomposition

model estimate for testcase n trained on dataset D

average target value for test case n

"Bias" term is the squared error of the average, over training datasets D, of the estimates.

~~Bias: average between prediction~~ and desired.

$$\left\langle \left\{ y(\mathbf{x}_n; D) - \bar{t}_n \right\}^2 \right\rangle_D = \boxed{\left\{ \left\langle y(\mathbf{x}_n; D) \right\rangle_D - \bar{t}_n \right\}^2}$$

$$+ \boxed{\left\langle \left\{ y(\mathbf{x}_n; D) - < y(\mathbf{x}_n; D) >_D \right\}^2 \right\rangle_D}$$

<. > means expectation over D

"Variance" term: variance over training datasets D, of the model estimate.

Regularization parameter affects the bias and variance terms

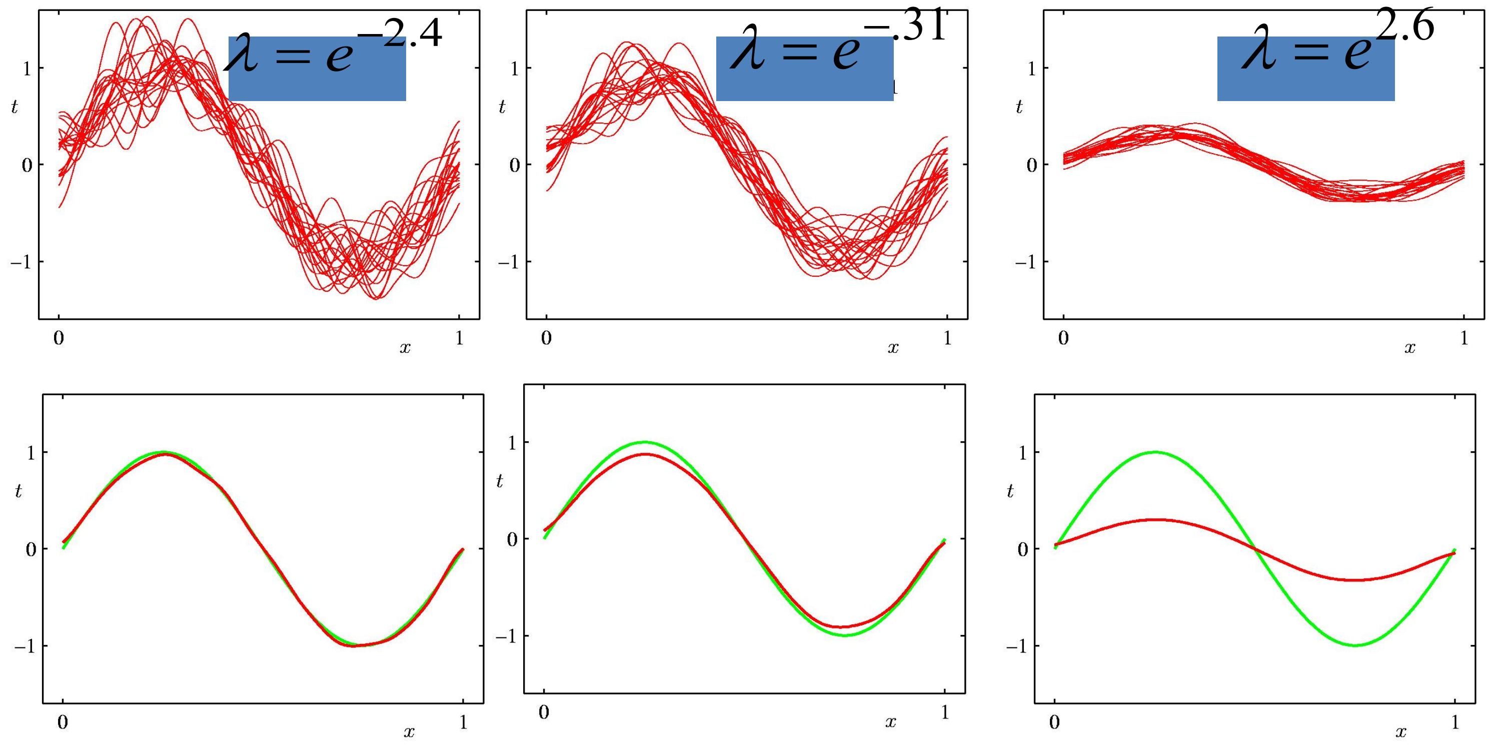
high variance

low variance

20 realizations

True model

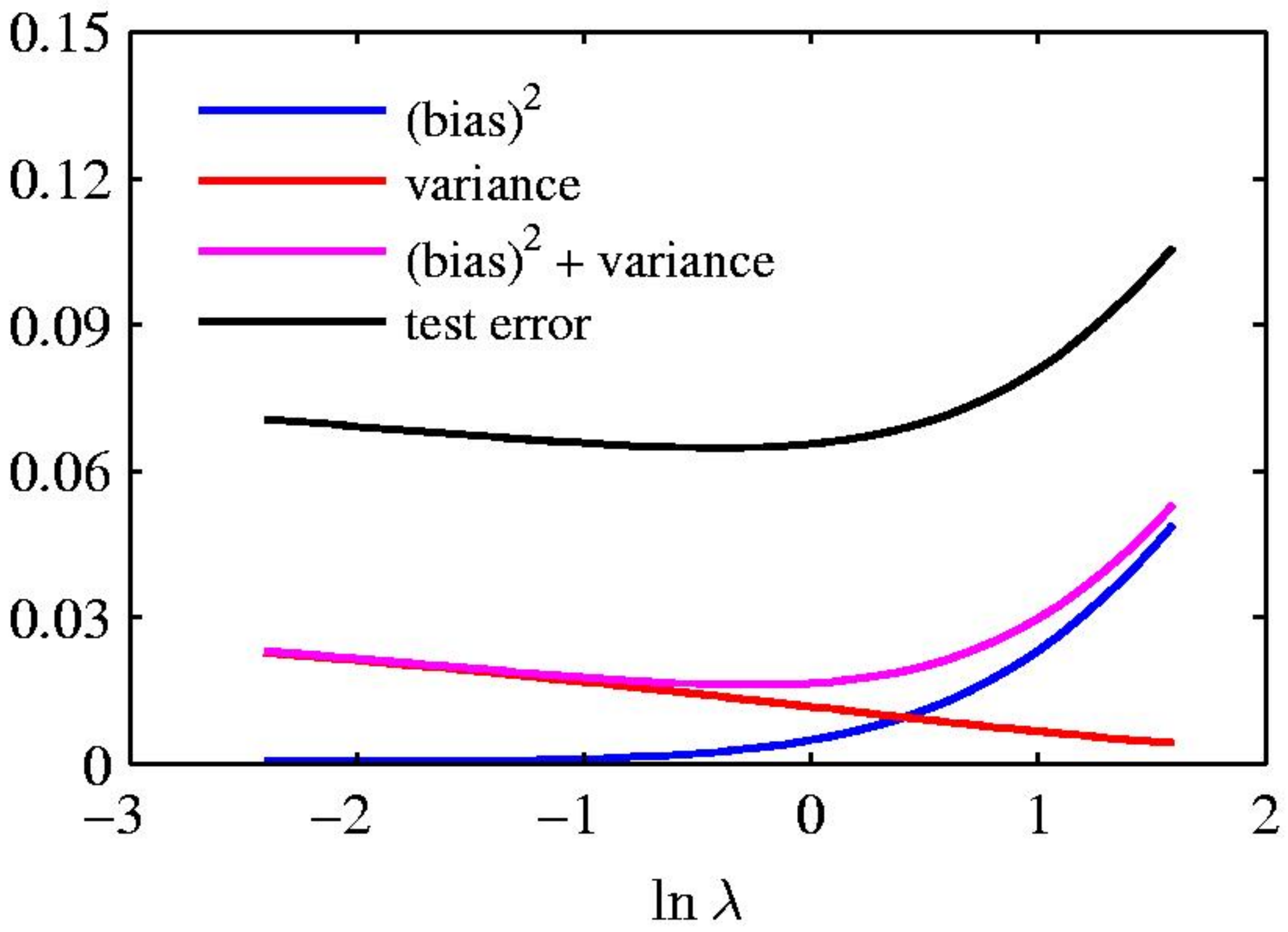
average



low bias

high bias

An example of the bias-variance trade-off



Beating the bias-variance trade-off

- We can reduce the variance term by averaging lots of models trained on different datasets.
  - This seems silly. If we had lots of different datasets it would be better to combine them into one big training set.
    - With more training data there will be much less variance.
- Weird idea: We can create different datasets by bootstrap sampling of our single training dataset.
  - This is called “bagging” and it works surprisingly well.
- If we have enough computation its better doing it Bayesian:
  - Combine the predictions of many models using the posterior probability of each parameter vector as the combination weight.

The Bayesian approach

- Consider a simple linear model that only has two parameters:

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

- It is possible to display the full posterior distribution over the two-dimensional parameter space.

- The likelihood term is a Gaussian, so if we use a Gaussian prior the posterior will be Gaussian:
  - This is a **conjugate prior.** It means that the prior is just like having already observed some data.

Bayesian Linear Regression (1)

- Define a conjugate prior over w

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0).$$

•Combining this with the likelihood function and using results for marginal and conditional Gaussian distributions, gives the posterior

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

- where

$$\begin{aligned}\mathbf{m}_N &= \mathbf{S}_N\left(\mathbf{S}_0^{-1}\mathbf{m}_0 + \beta\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}\right)\\ \mathbf{S}_N^{-1} &= \mathbf{S}_0^{-1} + \beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}.\end{aligned}$$

Bayesian Linear Regression (2)

- A common choice for the prior is

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$$

•for which

$$\begin{aligned} \mathbf{m}_N &= \beta \mathbf{S}_N \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t} \\ \mathbf{S}_N^{-1} &= \alpha \mathbf{I} + \beta \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi}. \end{aligned}$$

Gaussian

variance of output noise

$$p(\mathbf{t} \mid \mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathrm{N}(t_n \mid \mathbf{w}^T \mathbf{x}_n, \beta^{-1})$$

$\leftarrow$ likelihood

$$p(\mathbf{w} \mid \alpha) = \mathrm{N}(\mathbf{w} \mid 0, \alpha^{-1}\mathbf{I})$$

$\leftarrow$ conjugate prior

inverse variance of prior

$$-\ln p(\mathbf{w} \mid \mathbf{t}) = \frac{\beta}{2} \sum_{n=1}^{N} (t_n - \mathbf{w}^T \mathbf{x}_n)^2 + \frac{\alpha}{2} \mathbf{w}^T \mathbf{w} + const$$

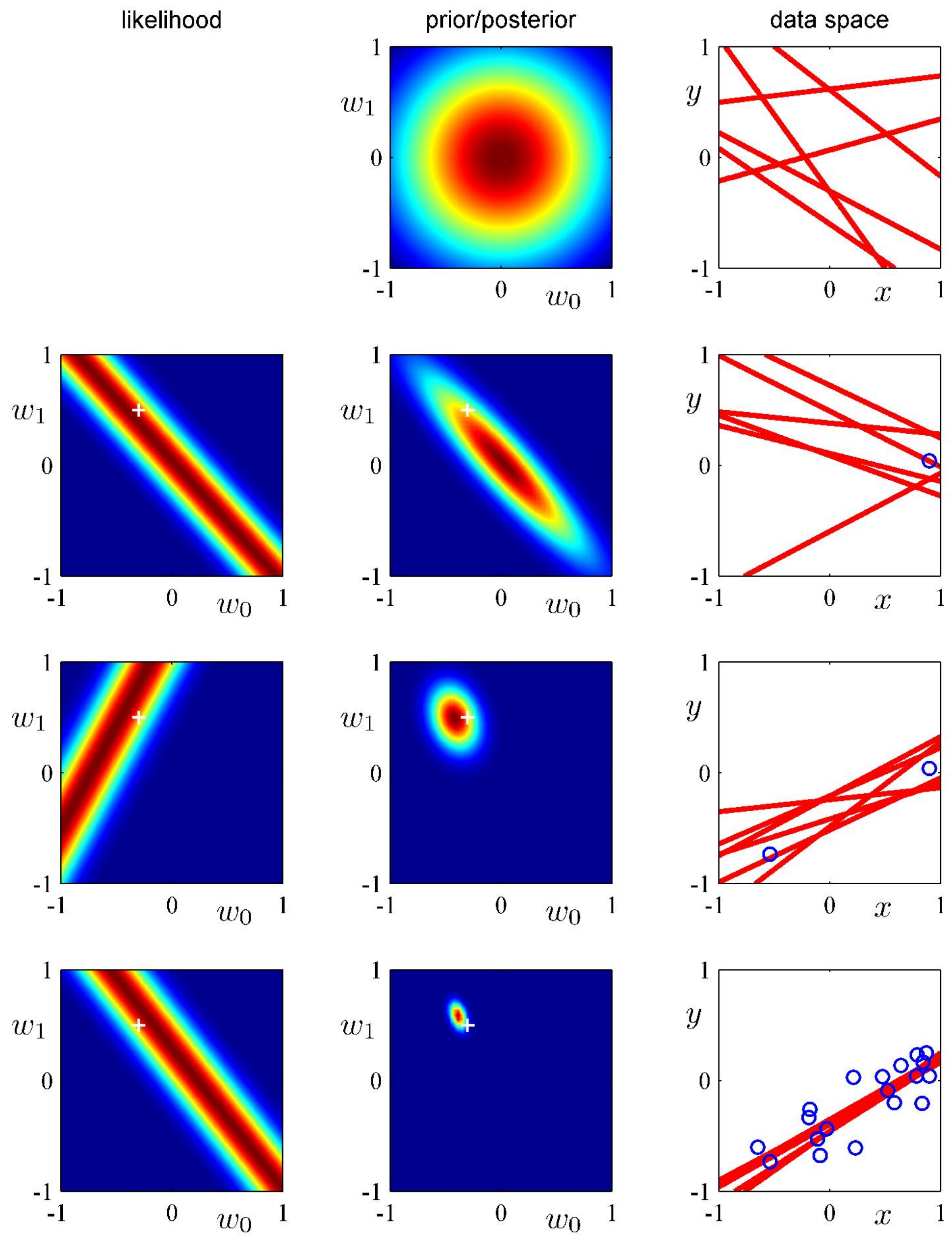
The Bayesian interpretation of the regularization parameter:

$$\lambda = \frac{\alpha}{\beta}$$

Bishop Fig 3.7

$$y = w_0 + w_1 x + N(0,0.2)$$

W_0=-0.3, w_1=0.5



With no data we sample lines from the prior.

With 20 data points, the prior has little effect

Using the posterior distribution

If we can afford the computation, we ought to average the predictions of all parameters weighted with the posterior distribution:

$$p(t_{test} \mid x_{test}, \alpha, \beta, \underset{\substack{\uparrow \\ \text{training} \\ \text{data}}}{D}) = \int p(t_{test} \mid x_{test}, \underset{\substack{\uparrow \\ \text{precision} \\ \text{of output} \\ \text{noise}}}{\beta}, \mathbf{w}) \; p(\mathbf{w} \mid \underset{\substack{\uparrow \\ \text{precision} \\ \text{of prior}}}{\alpha}, \beta, D) \, d\mathbf{w}$$

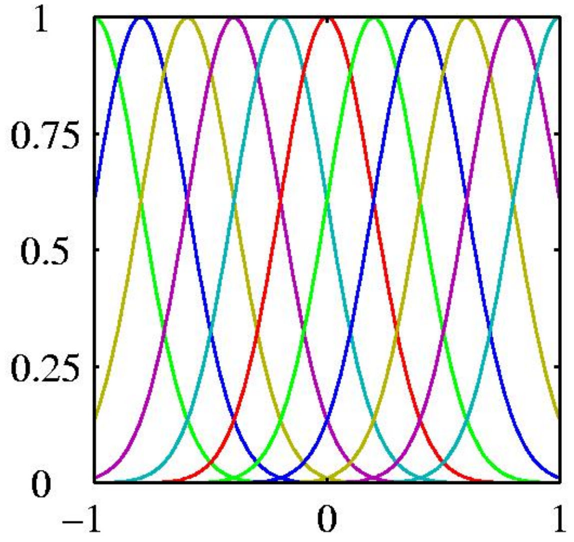
Predictive Distribution (1)

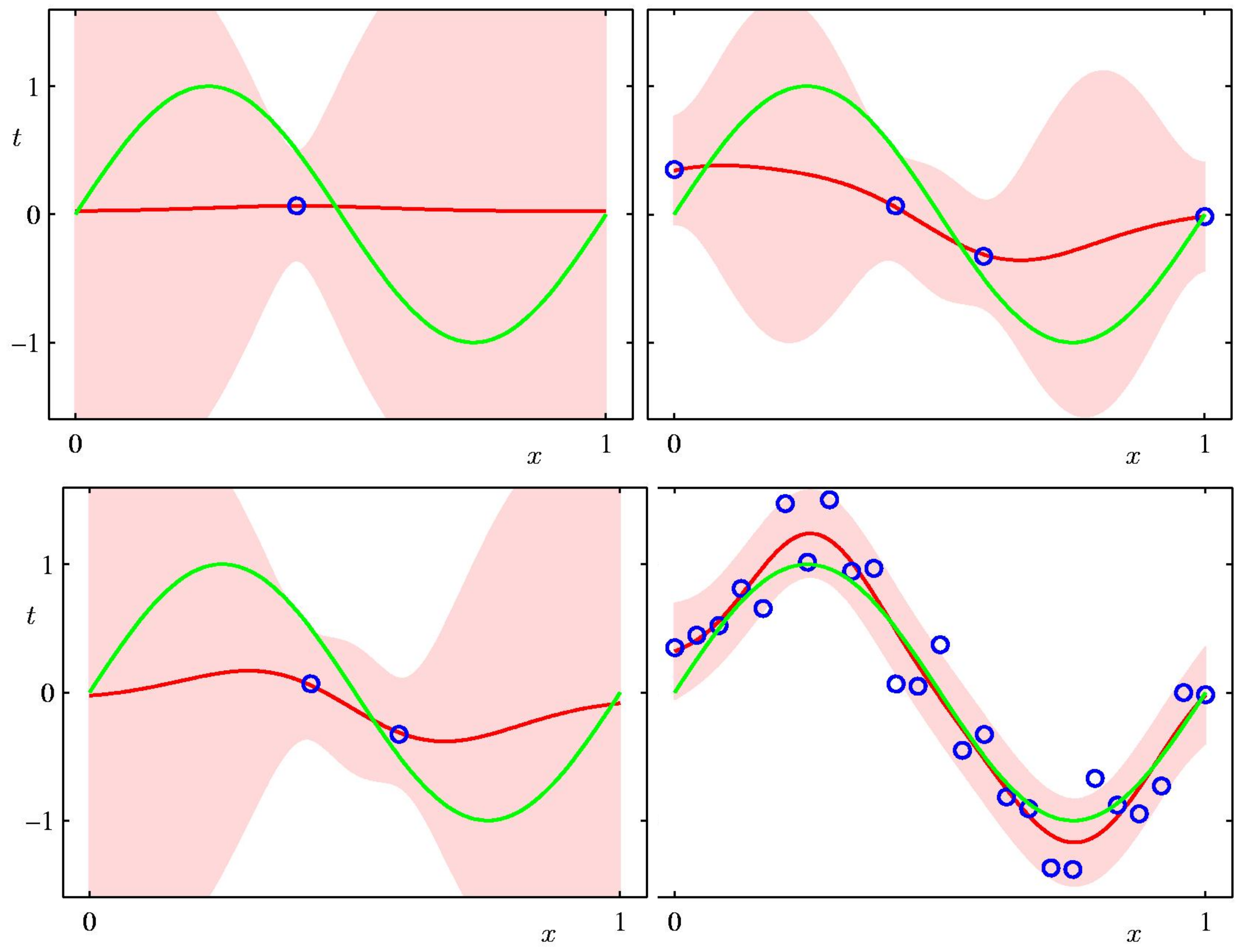
- Predict t for new values of x by integrating over w:
- where

$$\begin{aligned} p(t|\mathbf{t},\alpha,\beta) &= \int p(t|\mathbf{w},\beta)p(\mathbf{w}|\mathbf{t},\alpha,\beta)\,\mathrm{d}\mathbf{w} \\ &= \mathcal{N}(t|\mathbf{m}_N^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}),\sigma_N^2(\mathbf{x})) \end{aligned}$$

$$\sigma_N^2(\mathbf{x}) = \frac{1}{\beta} + \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\mathbf{S}_N\boldsymbol{\phi}(\mathbf{x}).$$

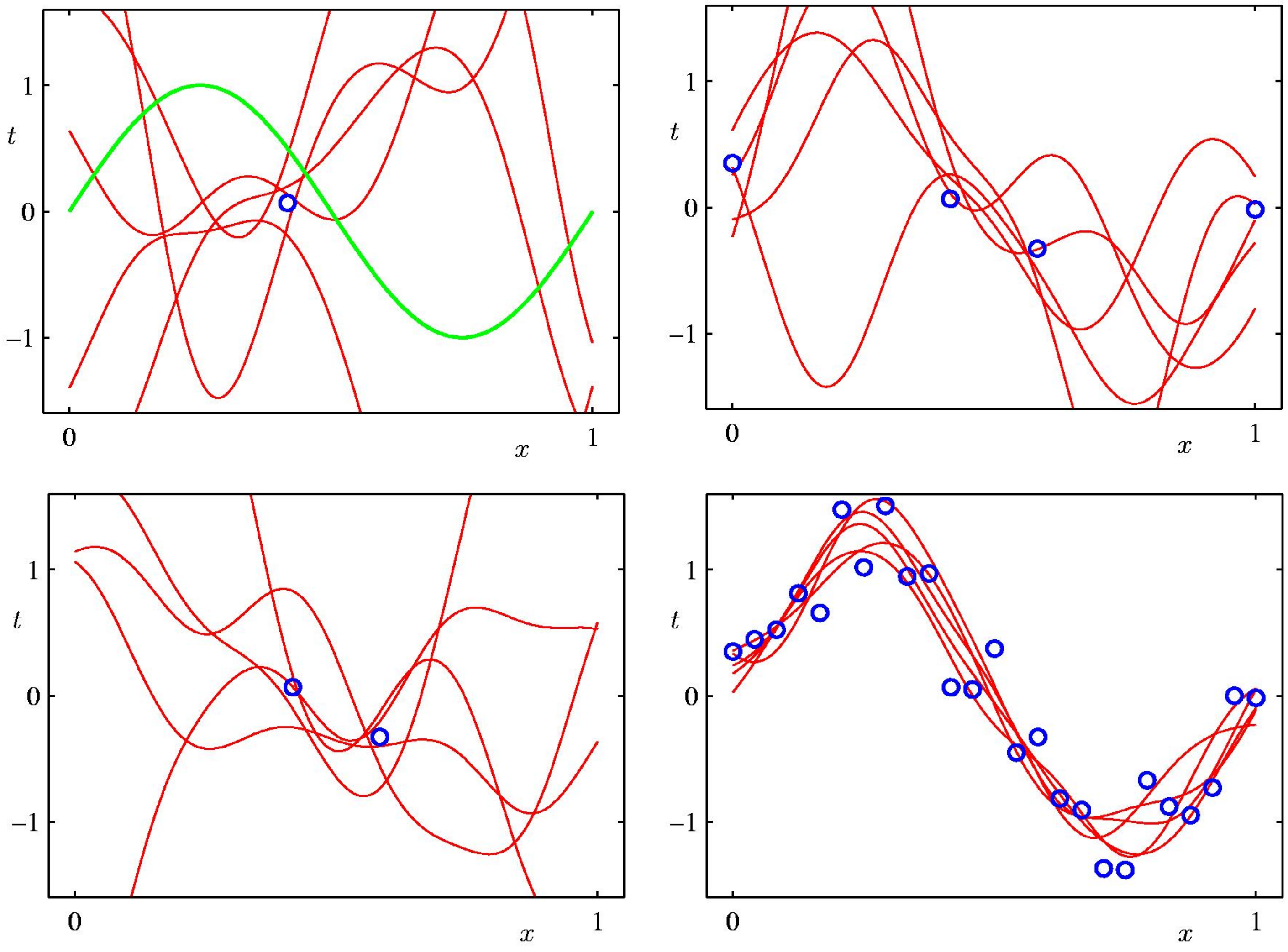
Predictive distribution for noisy sinusoidal data modeled by linear combining 9 radial basis functions.





A way to see the covariance of predictions for different values of x

We sample models at random from the posterior and show the mean of the each model's predictions



Bayesian model comparison

- We usually need to decide between many different models:
  - Different numbers of basis functions
  - Different types of basis functions
  - Different strengths of regularizers
- The frequentist way to decide between models is to hold back a validation set and pick the model that does best on the validation data.
  - This gives less training data. We can use a small validation set and evaluate models by training many different times using different small validation sets. But this is tedious.
- The Bayesian alternative is to use all of the data for training each model and to use the “**evidence**” to pick the best model (or to average over models).
- The **evidence is the marginal likelihood** with the parameters integrated out.

Definition of the evidence

The evidence is the normalizing term in the expression for the posterior probability of a weight vector given a dataset and a model class

$$p(\mathbf{w} \mid D, M_i) = \frac{p(\mathbf{w} \mid M_i)\, p(D \mid \mathbf{w}, M_i)}{p(D \mid M_i)}$$

$$p(D \mid M_i) = \int p(D \mid \mathbf{w}, M_i)\;\; p(\mathbf{w} \mid M_i)\, d\mathbf{w}$$

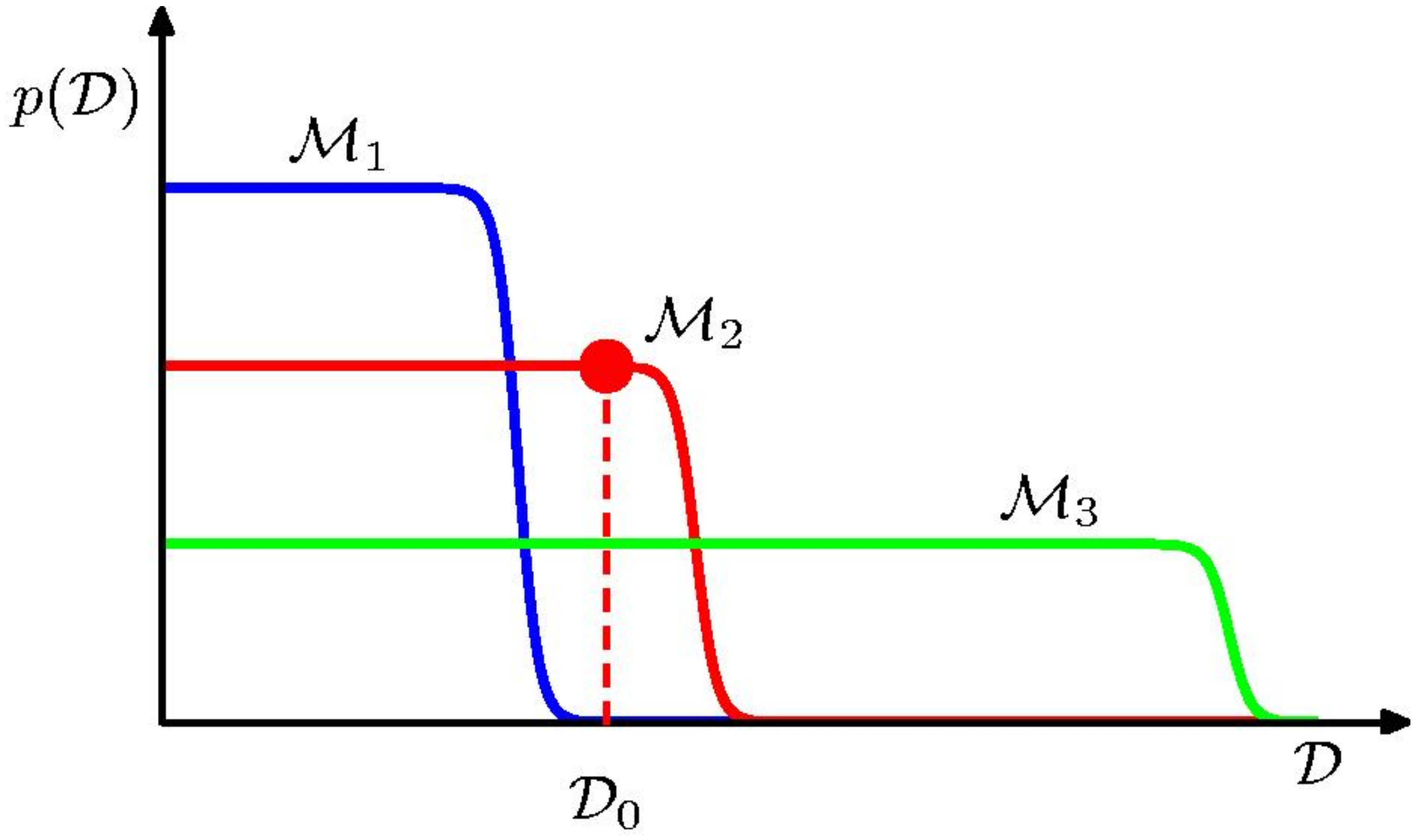
Using the evidence

- Now we use the evidence for a model class in exactly the same way as we use the likelihood term for a particular setting of the parameters
  - The evidence gives a posterior distribution over model classes, provided we have a prior.

$$p(M_i \mid D) \propto p(M_i)\, p(D \mid M_i)$$

  - For simplicity in making predictions we often pick the model class with the highest posterior probability. This is called model selection.
    - But we should still average over the parameter vectors for that model class using the posterior distribution.

How the model complexity affects the evidence



Increasingly complicated data →

Determining the hyperparameters that specify the prior variance and the variance of the output noise.

- Ideally, when making a prediction, we integrate out hyperparameters, just like we integrate out the weights
  - But this is infeasible even when everything is Gaussian.
- Empirical Bayes (also called the evidence approximation) means integrating out the parameters but maximizing over the hyperparameters.
  - Its more feasible and often works well.
  - It creates ideological disputes.

Empirical Bayes

target and input on test case

precision of output noise

precision of prior

training data

$$p(t \mid \mathbf{x}, D) = \int \int \int p(t \mid \mathbf{x}, \beta, \mathbf{w}) \; p(\mathbf{w} \mid \alpha, \beta, D) \; p(\alpha, \beta \mid D) \; d\mathbf{w} \; d\alpha \; d\beta$$

- The equation above is the right predictive distribution (assuming we do not have hyperpriors for alpha and beta).
- The equation below is a more tractable approximation that works well if the posterior distributions for alpha and beta are highly peaked (so the distributions are well approximated by their most likely values)

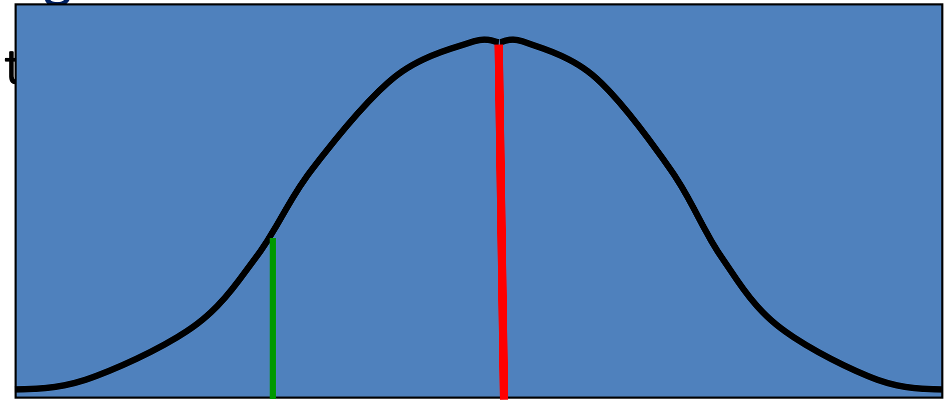
point estimates of alpha and beta that maximize the evidence

$$p(t \mid \mathbf{x}, D) \approx p(t \mid \mathbf{x}, D, \hat{\alpha}, \hat{\beta}) = \int p(t \mid \mathbf{x}, \hat{\beta}, \mathbf{w}) \; p(\mathbf{w} \mid \hat{\alpha}, \hat{\beta}, D) \; p(\hat{\alpha}, \hat{\beta} \mid D) \; d\mathbf{w} \; d\alpha \; d\beta$$

- OLD

When is minimizing the squared error equivalent to Maximum Likelihood Learning?

Minimizing the squared residuals is equivalent to maximizing the log probability of the correct answer under a Gaussian centered at the model's guess.



t = correct answer

y = model's estimate of most probable value

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

$$p(t_n \mid y_n) = p(y_n + noise = t_n \mid \mathbf{x}_n, \mathbf{w}) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(t_n - y_n)^2}{2\sigma^2}}$$

$$-\log p(t_n \mid y_n) = \log \sqrt{2\pi} + \log \sigma + \frac{(t_n - y_n)^2}{2\sigma^2}$$

can be ignored if sigma is fixed

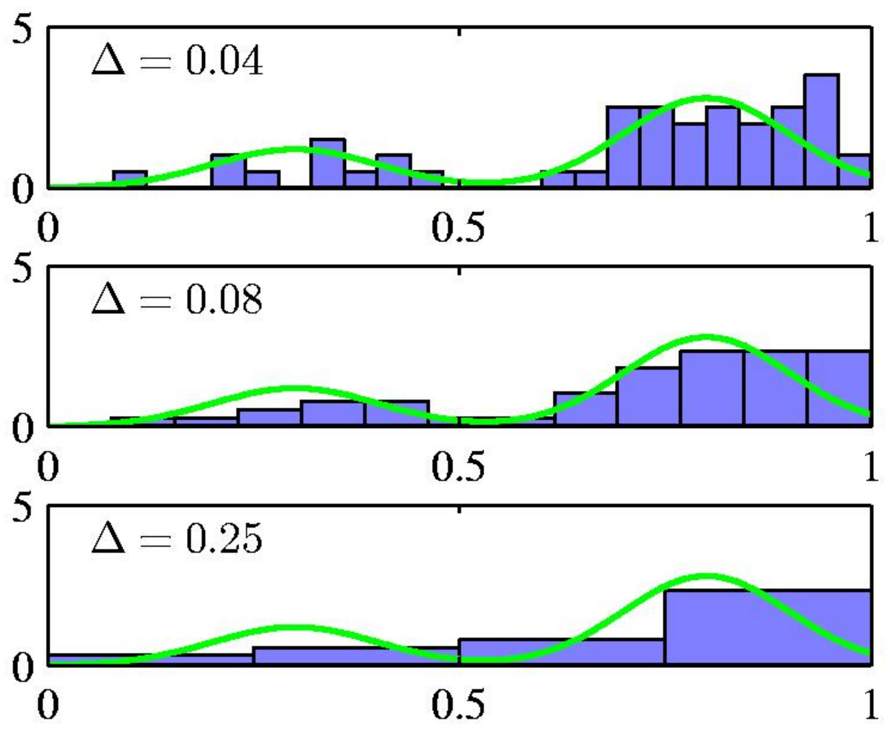
can be ignored if sigma is same for every case

Nonparametric Methods (2)

Histogram methods partition the data space into distinct bins with widths ¢$_i$ and count the number of observations, n_i, in each bin.

$$p_i = \frac{n_i}{N\Delta_i}$$

- Often, the same width is used for all bins, $\Delta_i = \Delta$.
- Δ acts as a *smoothing* parameter.
- In a D-dimensional space, using M bins in each dimension will require **M^D** bins! => it only work for marginals.



Nonparametric Methods (3)

•Assume observations drawn from a density p(x) and consider a small region R containing x such that

$$P = \int_{\mathcal{R}} p(\mathbf{x})\,\mathrm{d}\mathbf{x}.$$

•The probability that K out of N observations lie inside R is Bin(KjN,P) and if N is large

$$K \simeq NP.$$

If the volume of R, V, is sufficiently small, p(x) is approximately constant over R and

$$P \simeq p(\mathbf{x})V$$

Thus

$$\boxed{p(\mathbf{x}) = \frac{K}{NV}.}$$

V small, yet K>0, therefore N large?

Nonparametric Methods (4)

Kernel Density Estimation: fix V, estimate K from the data. Let R be a hypercube centred on x and define the kernel function (Parzen window)

$$k((\mathbf{x}-\mathbf{x}_n)/h) = \begin{cases} 1, & |(x_i - x_{ni})/h| \leqslant 1/2, \qquad i = 1, \ldots, D, \\ 0, & \text{otherwise.} \end{cases}$$

- It follows that
- $$K = \sum_{n=1}^{N} k\left(\frac{\mathbf{x}-\mathbf{x}_n}{h}\right)$$ and hence $$p(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{h^D}k\left(\frac{\mathbf{x}-\mathbf{x}_n}{h}\right).$$

Nonparametric Methods (5)

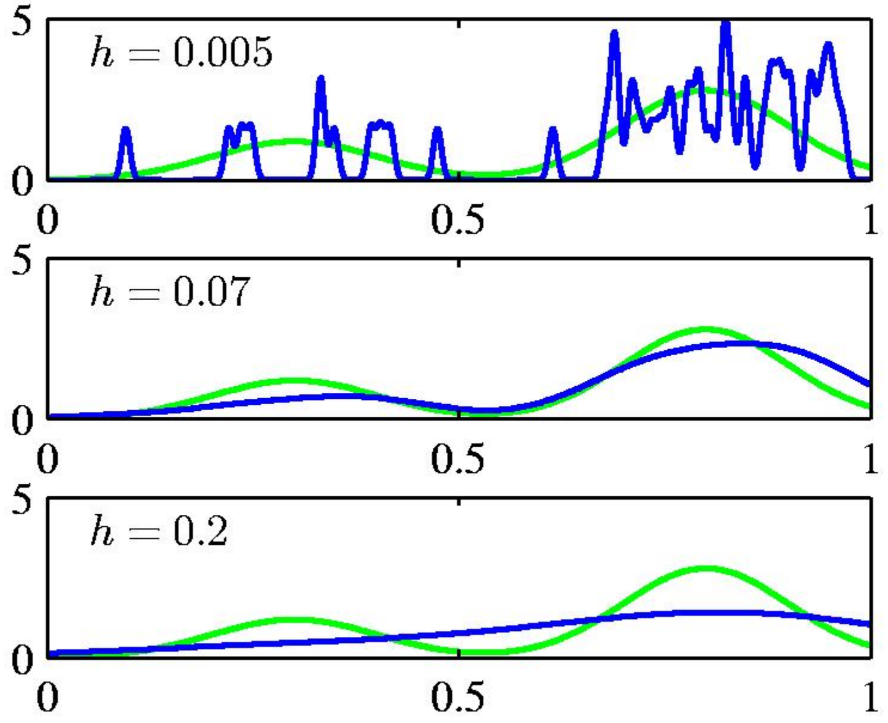
To avoid discontinuities in p(x), use a smooth kernel, e.g. a Gaussian

$$p(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{(2\pi h^2)^{D/2}}\exp\left\{-\frac{\|\mathbf{x}-\mathbf{x}_n\|^2}{2h^2}\right\}$$

Any kernel such that

$$\begin{aligned} k(\mathbf{u}) &\geqslant 0, \\ \int k(\mathbf{u})\,\mathrm{d}\mathbf{u} &= 1 \end{aligned}$$

will work.

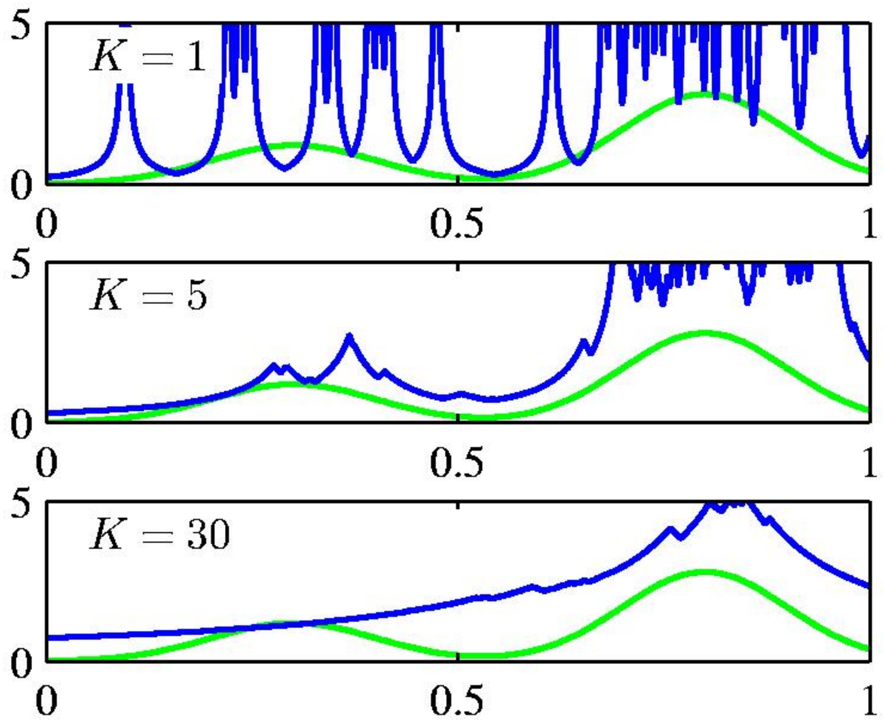


h acts as a smoother.

Nonparametric Methods (6)

Nearest Neighbour Density Estimation: fix K, estimate V from the data. Consider a hypersphere centred on x and let it grow to a volume, V ?, that includes K of the given N data points. Then

$$p(\mathbf{x}) \simeq \frac{K}{NV^\star}.$$



K acts as a smoother.

K-Nearest-Neighbours for Classification (1)

- Given a data set with N_k data points from class C_k and $\sum_k N_k = N$, we have

$$p(\mathbf{x}) = \frac{K}{NV}$$

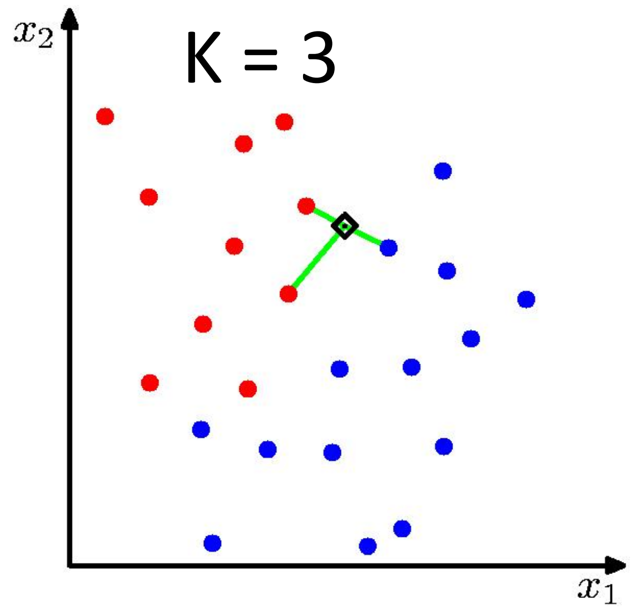
- and correspondingly

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{K_k}{N_k V}.$$

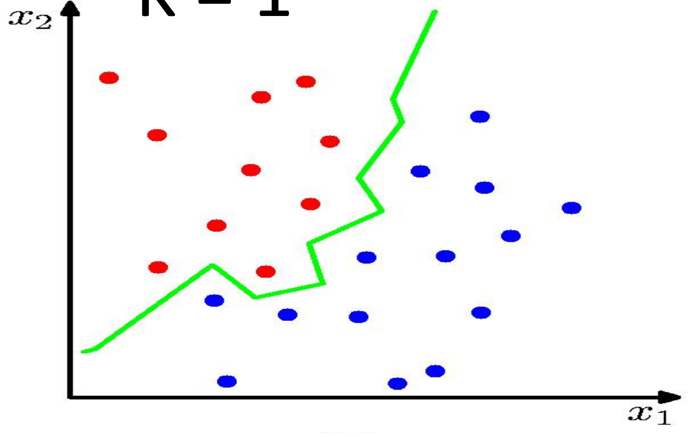
- Since $p(\mathcal{C}_k) = N_k/N$, Bayes' theorem gives

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})} = \frac{K_k}{K}.$$

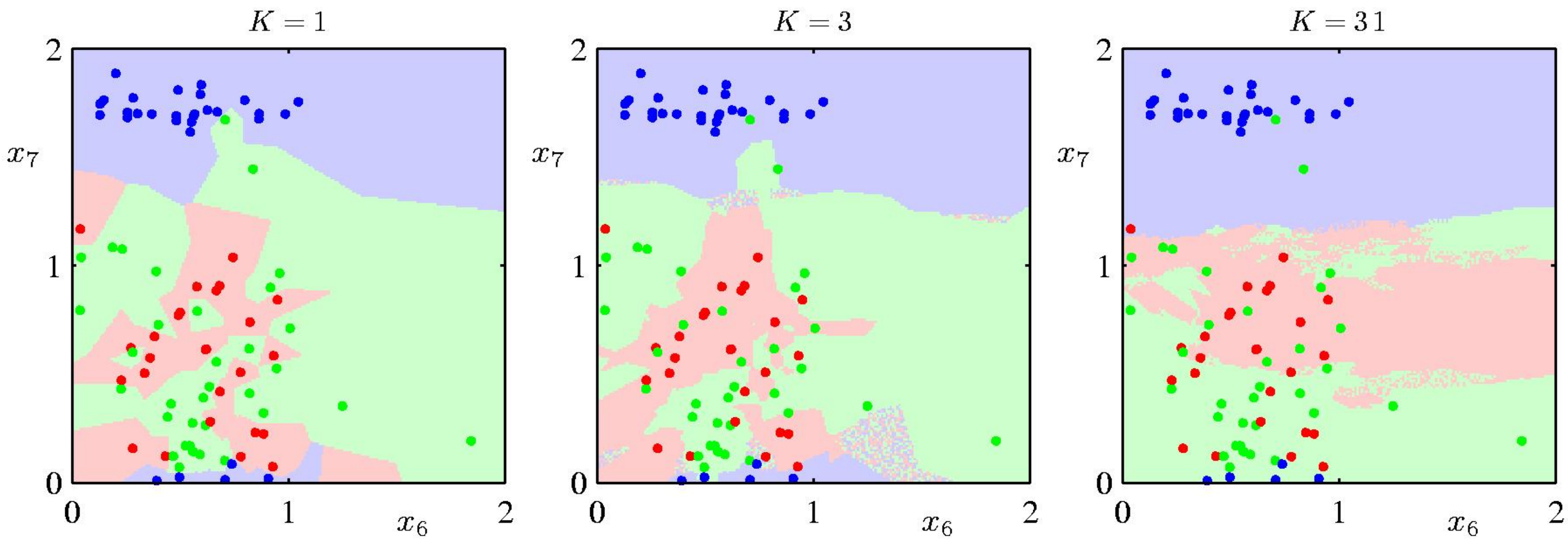
K = 3



K = 1



K-Nearest-Neighbours for Classification (3)



- K acts as a smother
- For $N \to \infty$, the error rate of the nearest-neighbour (K=1) classifier is never more than twice the optimal error (from the true conditional class distributions).

Minimizing squared error

$$y = \mathbf{w}^T \mathbf{x}$$

$$error = \sum_n (t_n - \mathbf{w}^T \mathbf{x}_n)^2$$

$$\mathbf{w}^* = \boxed{(\mathbf{X}^T \mathbf{X})^{-1}} \mathbf{X}^T \boxed{\mathbf{t}} \leftarrow$$

vector of target values

↑ optimal weights

↑ inverse of the covariance matrix of input vectors

↑ the transposed design matrix has one input vector per column

The loss function

- Fitting a model to data is typically done by finding the parameter values that minimize some loss function.
- There are many possible loss functions. What criterion should we use for choosing one?
  - Choose one that makes the math easy (squared error)
  - Choose one that makes the fitting correspond to maximizing the likelihood of the training data given some noise model for the observed outputs.
  - Choose one that makes it easy to interpret the learned coefficients (easy if mostly zeros)
  - Choose one that corresponds to the real loss on a practical application (losses are often asymmetric)

SKIP

Linear models

- It is mathematically easy to fit linear models to data.
  - We can learn a lot about model-fitting in this relatively simple case.
- There are many ways to make linear models more powerful while retaining their nice mathematical properties:
  - Using non-linear, non-adaptive basis functions, we get generalised linear models that learn non-linear mappings from input to output but are linear in their parameters – only the linear part of the model learns.
  - Using kernel methods we can handle expansions of the raw data that use a huge number of non-linear, non-adaptive basis functions.
  - Using large margin kernel methods we can avoid overfitting even when with huge numbers of basis functions.
- But linear methods will not solve most AI problems.
  - They have fundamental limitations.

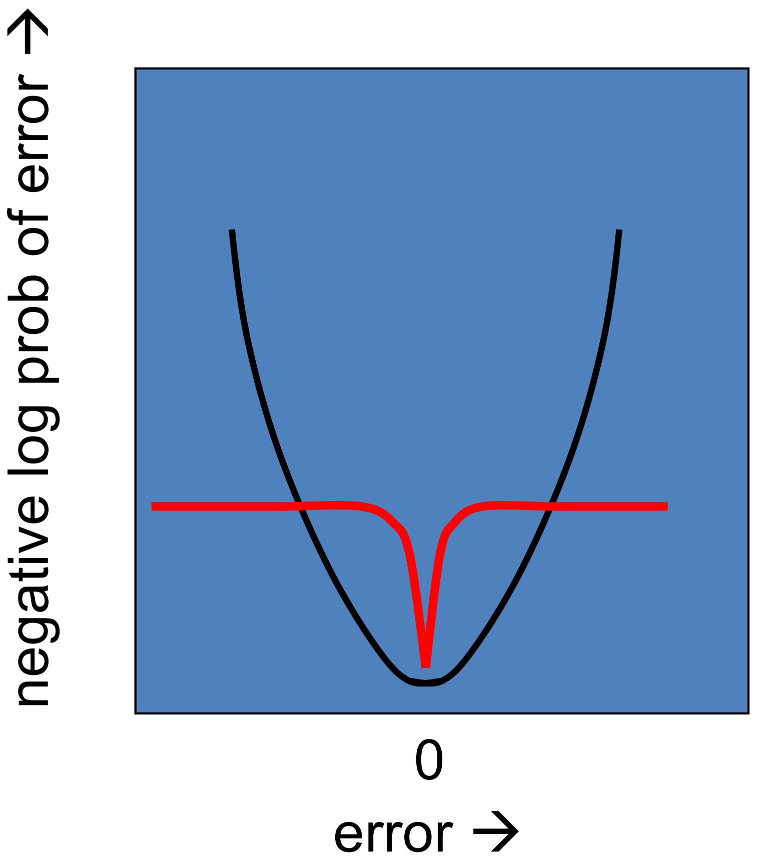
SKIP

An example where minimizing the squared error gives terrible estimates

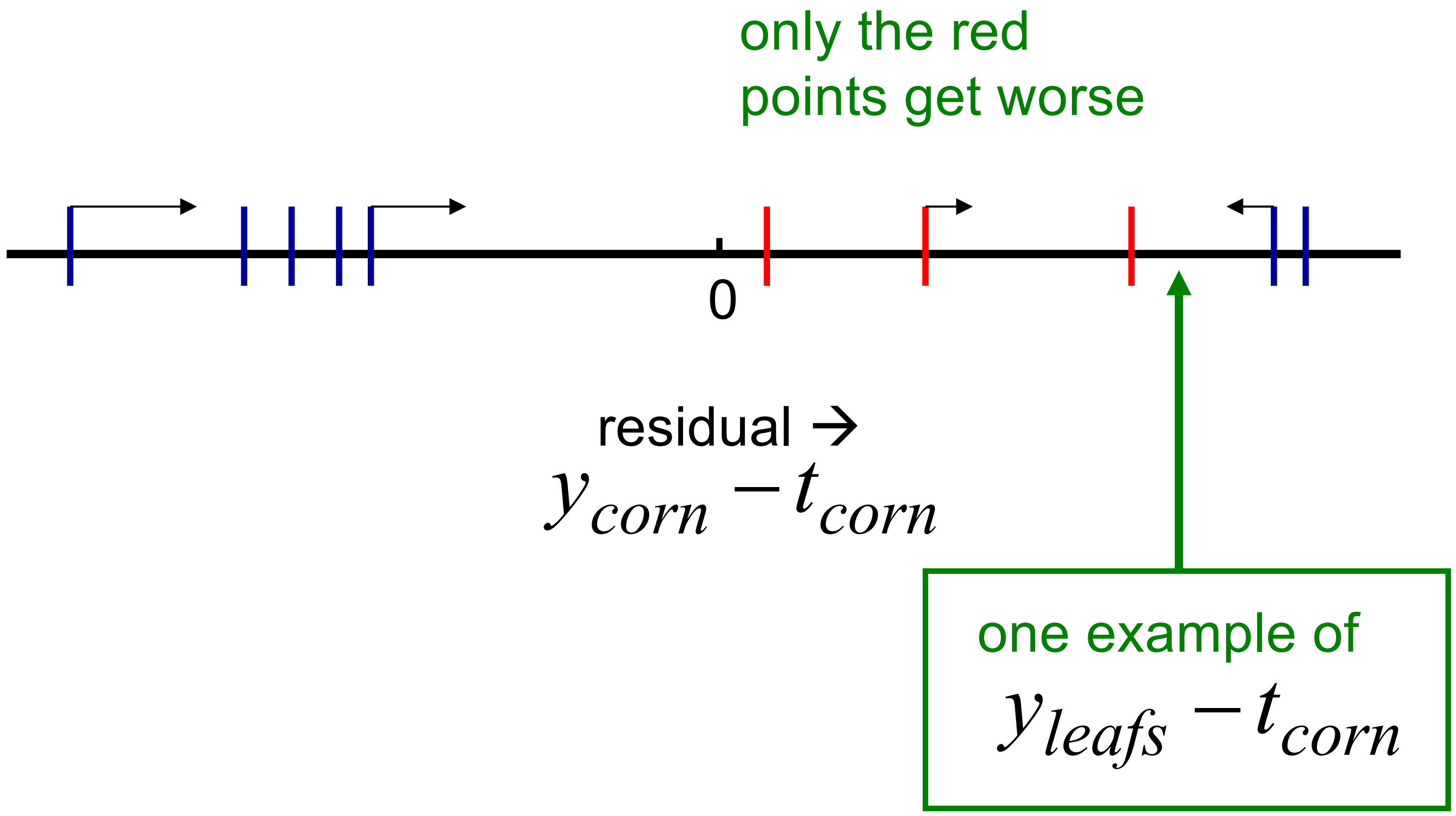
- Suppose we have a network of 500 computers and they all have slightly imperfect clocks.
- After doing statistics 101 we decide to improve the clocks by averaging all the times to get a least squares estimate
  - Then we broadcast the average to all of the clocks.

- Problem: The probability of being wrong by ten hours is more than one hundredth of the probability of being wrong by one hour. In fact, its about the same!

SKIP



Why shrinkage helps



SKIP

If we move all the blue residuals towards the green arrow by an amount proportional to their difference, we are bound to reduce the average squared magnitudes of the residuals. So if we pick a blue point at random, we reduce the expected residual.