Machine Learning 1

Lecture 03 - Linear Methods for Regression

Patrick Forré

1 Linear Methods for Regression

Linear Basis Function Model

- Training data: $D = (x_1, \ldots, x_N)^T$ with targets $T = (t_1, \ldots, t_N)^T$, where every $x_i \in \mathbb{R}^D$ is a D-dimensional vector $x_i = (x_{i,1}, \ldots, x_{i,D})^T$.
- Fix a number M and choose basis functions/"features" of x: $(\phi_0(x), \ldots, \phi_{M-1}(x))^T =: \phi(x)$, with $\phi_0 \equiv 1$.
- Model functions with parameters $w = (w_0, \ldots, w_{M-1}) \in \mathbb{R}^M$:

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

- Sum-of-squares error function:

$$E(D, T, w) := \frac{1}{2} \sum_{i=1}^{N} (t_i - y(x_i, w))^2.$$

- Unique minimizer (if existent): $w_{\mathrm{ML}} = (\Phi^T \Phi)^{-1} \Phi^T T$, with $N \times M$-matrix Φ with entries $\Phi_{ik} = \phi_k(x_i)$.

Example: Polynomial Regression

- We now want to see how good the naive approach ("minimizing the training error") works in practice, measured by the test error (by root-mean-squared error, RMSE).
- We generate the function $h(x) = \sin(2\pi x)$ and add Gaussian noise ϵ:

$$t = \sin(2\pi x) + \epsilon, \qquad \epsilon \sim \mathcal{N}(e|0, \sigma^2).$$

- We will take $N = 10$ data points $(x_1, \ldots, x_{10})$ and compute $(t_1, \ldots, t_{10})$.
- Goal: Try to infer the function h from the data points by 1-dimensional polynomial regression of degree M with $w \in \mathbb{R}^{M+1}$:

$$y(x, w) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M.$$

- We will minimize the training error, i.e. computing the Maximum Likelihood / Least-Sum-of-Square solution w_{ML}.

Example: Polynomial Regression

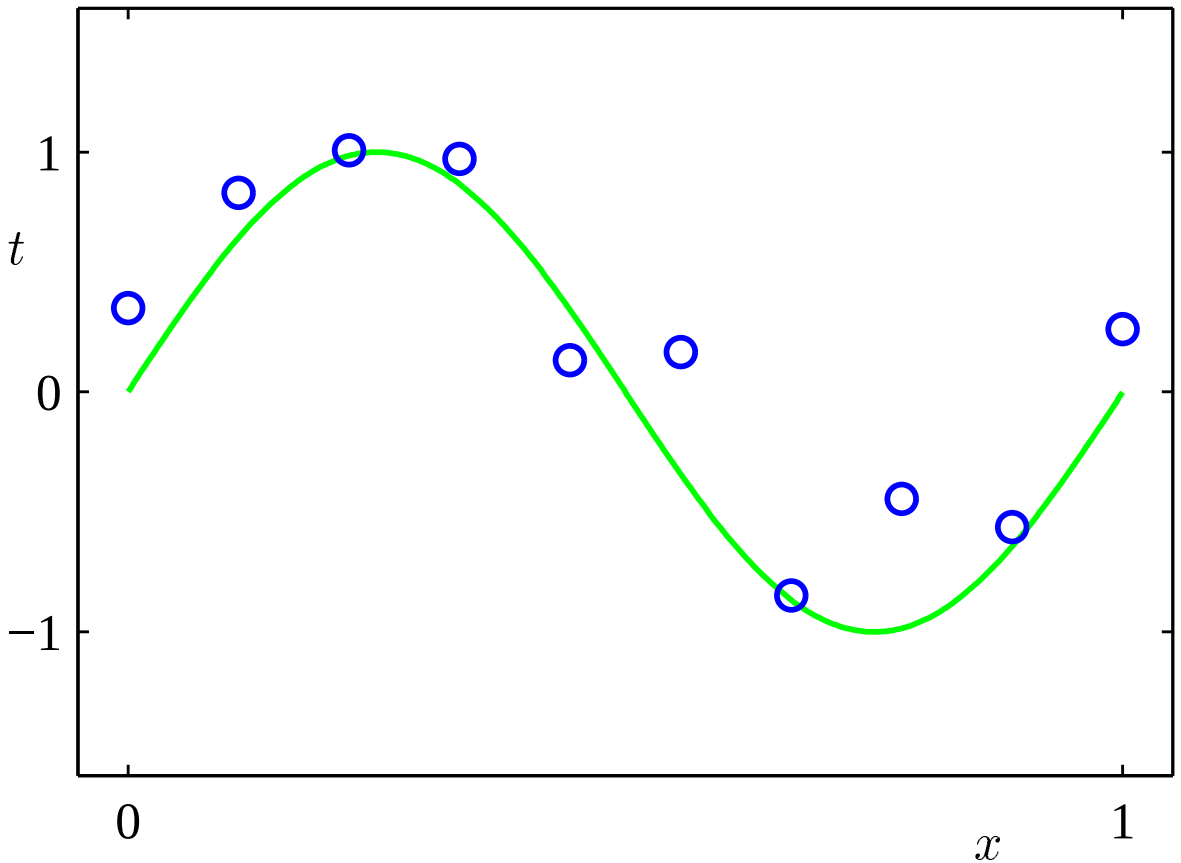

Figure: "Unknown" sinoidal curve ($h(x) = \sin(2\pi x)$) and observed data points ($t = h(x) + \epsilon$) (Bishop 1.2)

Example: Underfitting and Overfitting

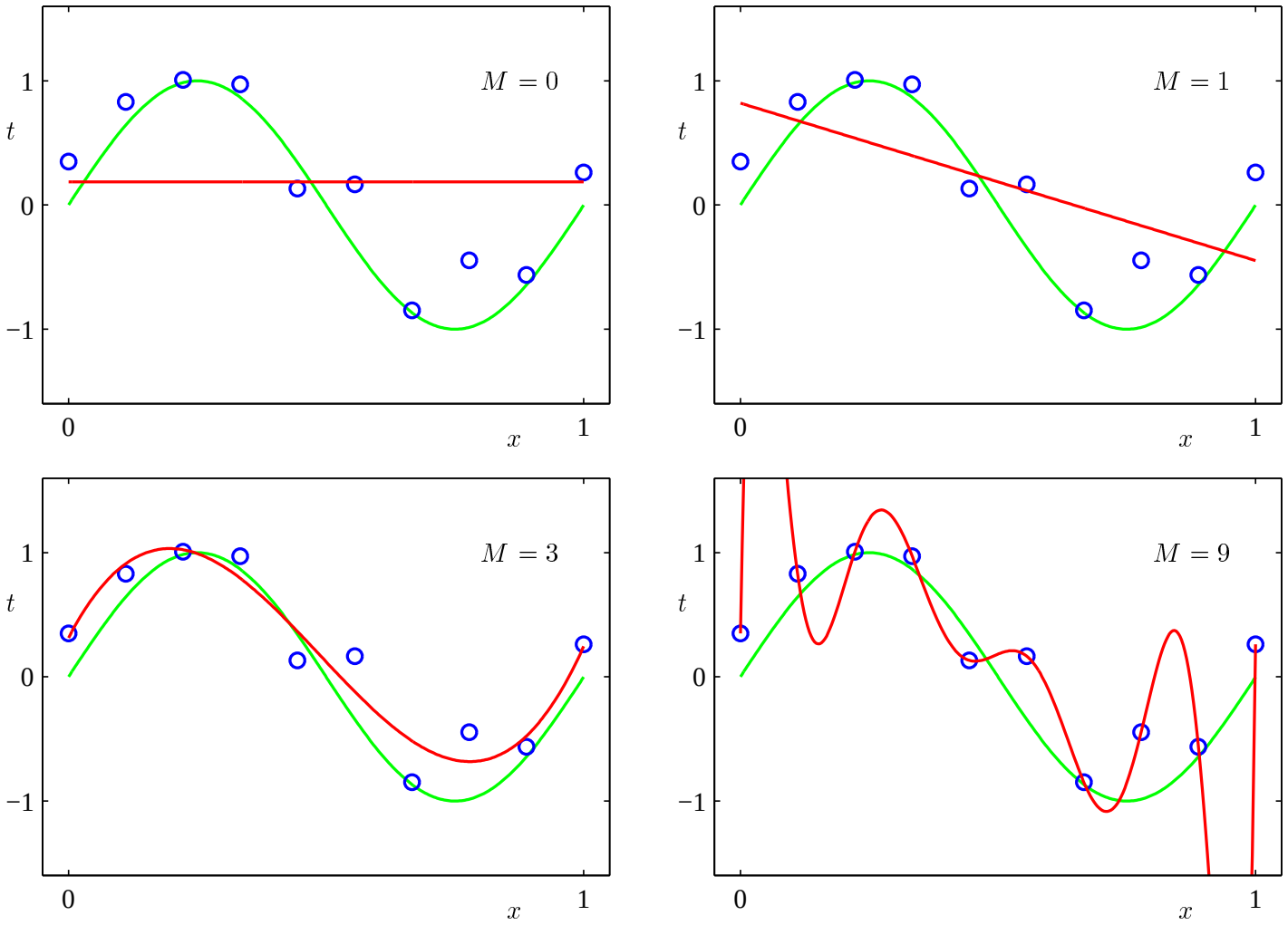

Table: Underfitting and Overfitting (Bishop 1.4)

Example: Training Error vs. Test Error by Complexity

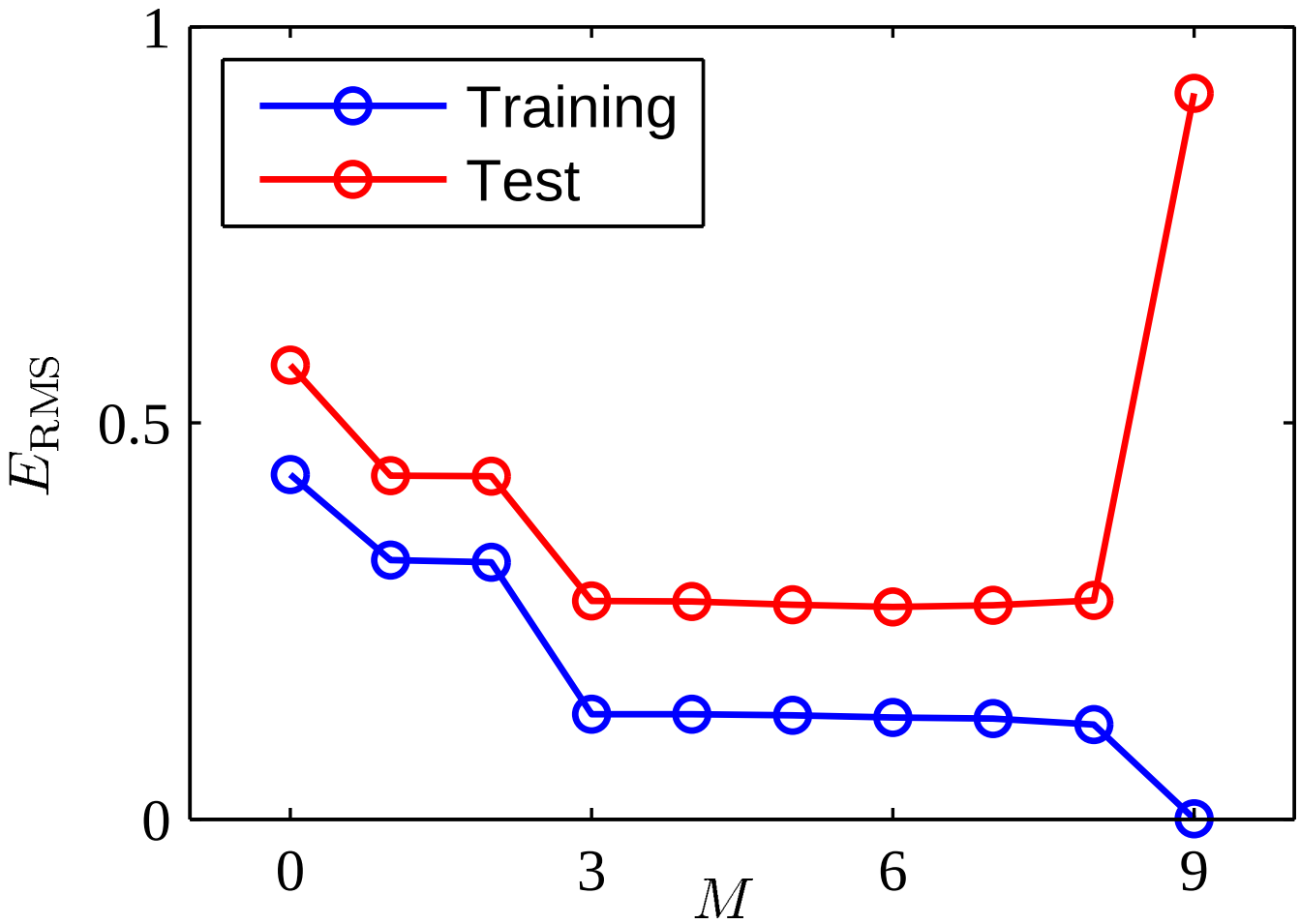

Figure: Training Error vs. Test Error by Complexity (Bishop 1.5)

Example: Size of Polynomial Coefficients vs. Overfitting

Polynomial Coefficients

	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Figure: Size of parameters indicates overfitting (Bishop Table 1.1)

Size of Coefficients vs. Overfitting

Question

- *Why does the model tend to overfit when the complexity M is big in comparison to the size of the training data N?*
- *And why is overfitting associated with relatively "big" coefficients?*

- If M is big then the model is flexible enough to fit the random noise terms of the N data points, resulting in overfitting:
- Overfitting means that the test error is much bigger than the training error.
- This means that on test data (x, t) the learned function

$$y(x, w) = w_0 + w_1\phi_1(x) + w_2\phi_2(x) + \cdots + w_{M-1}\phi_{M-1}(x)$$

  "hugely" differs from the expected target value t.
- But this means that some of the functions $w_i\phi_i(x)$ must tend to overshoot in the sum.
- So if ϕ_i is normalized then w_i must be "big".

Example: Sample Size vs. Overfitting

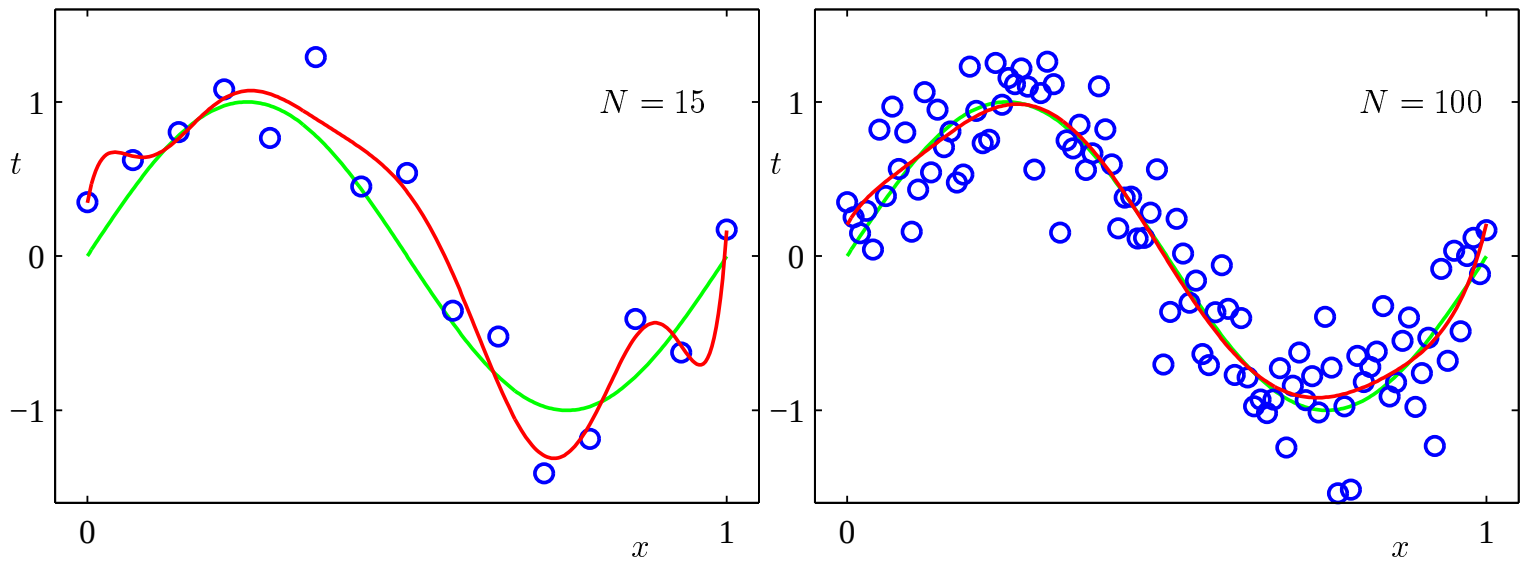

Figure: Polynomial fit of degree $M = 9$ and different number of data points N. Increase of N reduces overfitting (Bishop 1.6)

Problems: Underfitting and Overfitting

- Underfitting: model not flexible/complex enough (M too low) to capture variability of true function f.

  Detection: both training and test error comparatively high.

  Possible solutions:
  - Increase parameter space $\mathcal{W}$, i.e. complexity M,
  - create additional basis functions / "features" ϕ_j of the data x,
  - measure new meaningful properties of the samples.
- Overfitting: model too flexible (M too big in comparison to number of observations N). It will start to model variance and noise instead of true underlying function.

  Detection: training error low, test error high.

  Possible solutions:
  - get more data (increase N).
  - decrease parameter space $\mathcal{W}$, i.e. lower complexity M,
  - penalize big parameters / coefficients w_i ("Shrinkage", "Weight Decay", "Regularization", "Bayesian Approach").

Regularization and Regularized Regression

- To avoid overfitting we introduce a new term in the error function to penalize for big parameters:
- Carefully choose a parameter $\lambda \geqslant 0$ and $q > 0$ for a data set $D' = (x'_1, \ldots, x'_L)$ with targets $T' = (t'_1, \ldots, t'_L)$ put

$$\begin{aligned} E_{\mathrm{RG}}(D', T', w) &:= E(D', T', w) + \frac{\lambda}{q}||w||_q^q \\ &= \frac{1}{2}\sum_{i=1}^{L}(t_i - y(x_i, w))^2 + \frac{\lambda}{q}\sum_{k=0}^{M-1}|w_k|^q. \end{aligned}$$

- Regularized Regression: Minimize $E_{\mathrm{RG}}(D_{\mathrm{tr}}, T_{\mathrm{tr}}, w)$ on the training set w.r.t. to w to get w_{RG}.
- $y(x, w_{\mathrm{RG}})$ might do worse on training data (in terms of RMSE) but is supposed to do better on test data than $y(x, w_{\mathrm{ML}})$.
- $q = 1$ is called Lasso- and $q = 2$ the Ridge Regularization.
- For the Linear Basis Function Model we stick to $q = 2$. The Ridge Regression then has the unique closed form minimizer:

$$w_{\mathrm{RG}} = (\lambda \mathbb{1} + \Phi^T\Phi)^{-1}\Phi^T T.$$

Example: Regularization

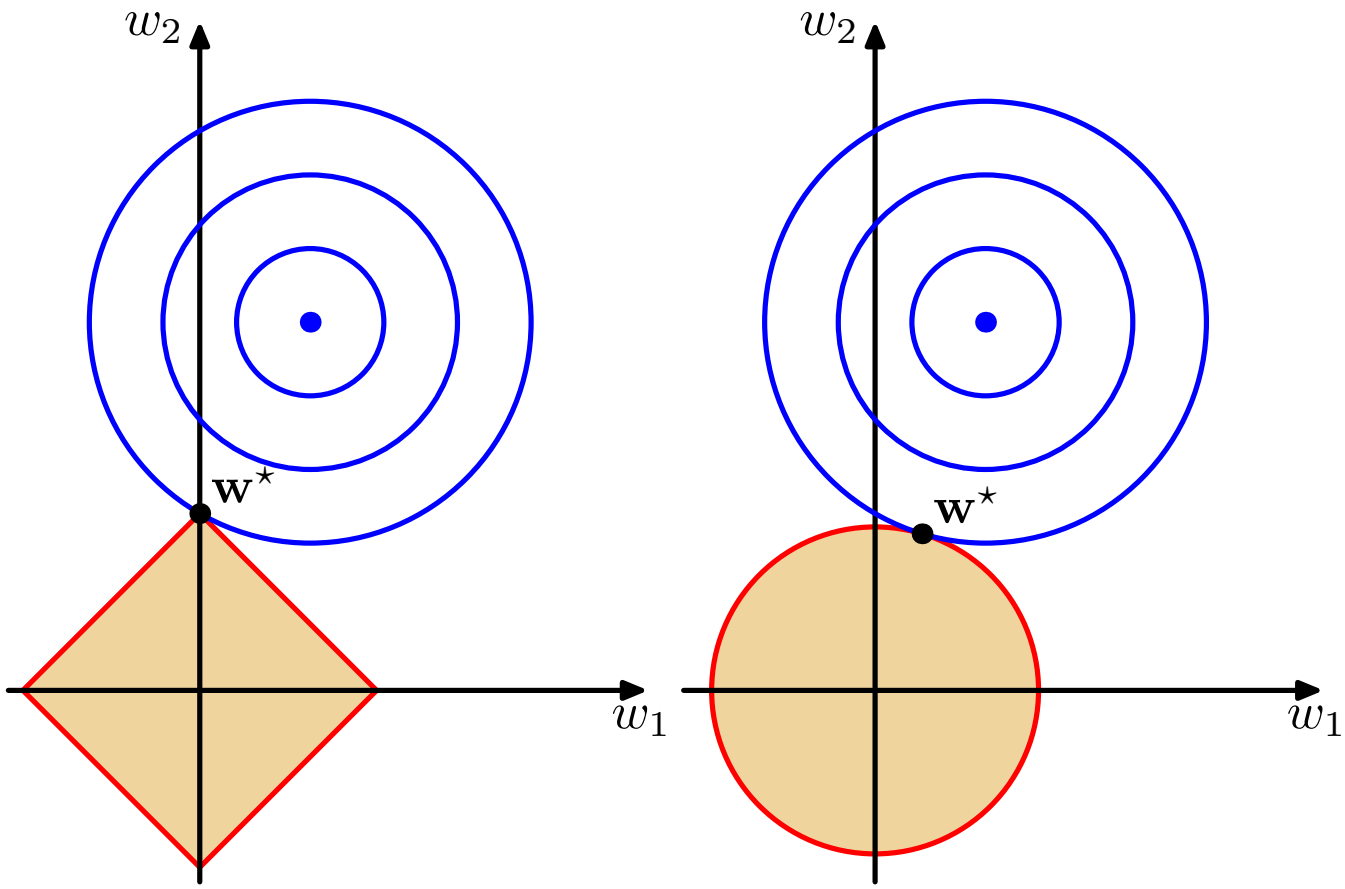

Figure: Minimizing $E_{\mathrm{RG}}(w)$ is equivalent to minimizing $E(w)$ under the constraint $||w||_q \leqslant \eta$. Small q (left: $q = 1$) lead to values w with lots of zero-entries, whereas bigger q (right: q=2) lead to more equally sized entries of w. (Bishop 3.4)

Example: Polynomial Ridge Regression

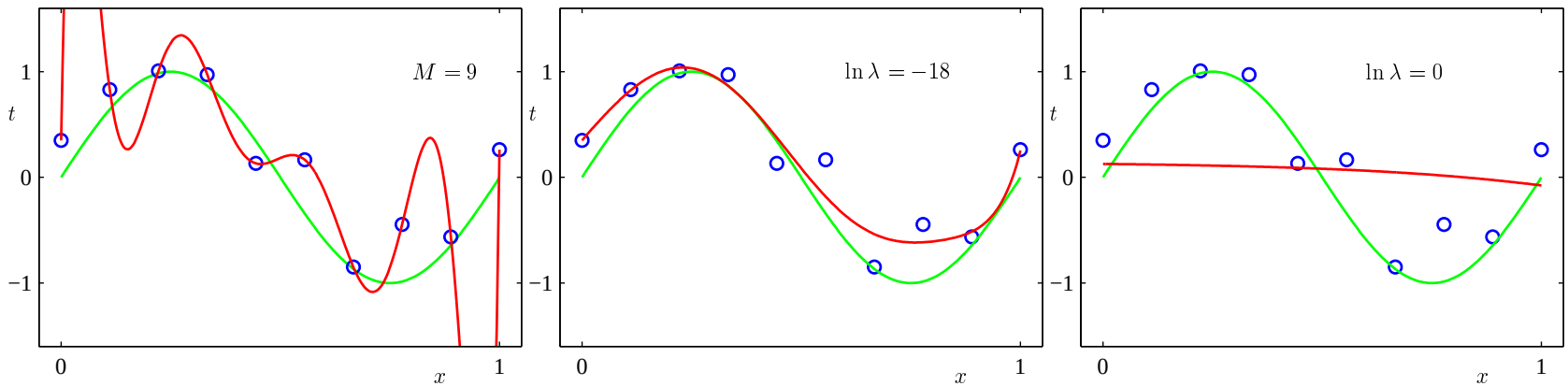

Figure: Penalized polynomial fit of degree $M = 9$ with different size of regularization parameter λ. The fit might get worse if λ is too big (underfitting) or too small (overfitting). Left: unregularized, i.e. $\ln \lambda = -\infty$. (Bishop 1.7)

Example: Training Error vs. Test Error with Regularization

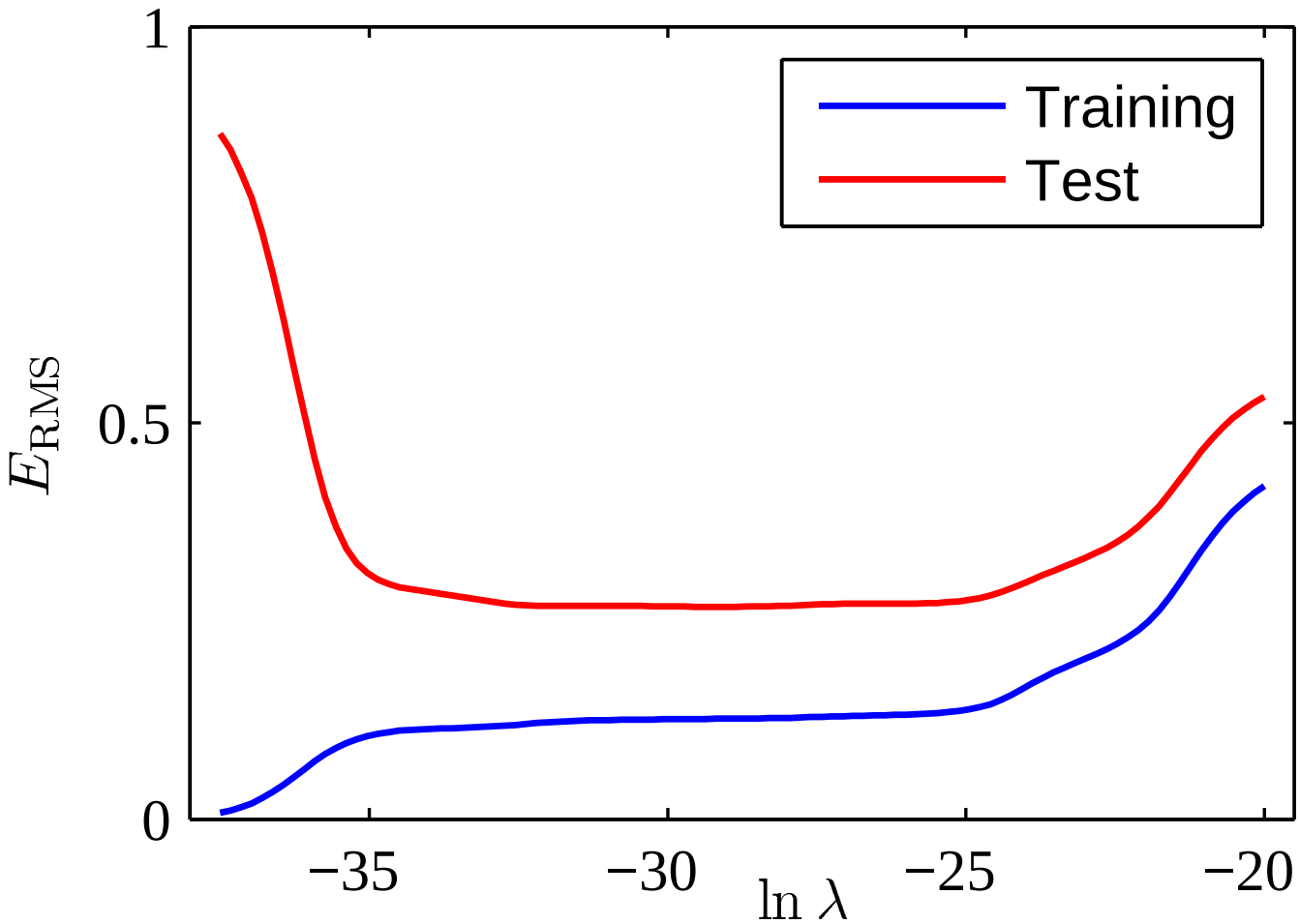

Figure: Training vs. test error by regularization parameter λ (Bishop 1.8)

Example: Size of Polynomial Coefficients with Regularization

Polynomial Coefficients

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

Figure: Coefficients of polynomial ridge regression of degree $M = 9$ by regularization parameter λ. $\ln \lambda = -\infty$ corresponds to the unregularized fit. Coefficients tend to get smaller with increasing λ. (Bishop Table 1.2)

Model Comparison and Model Selection

Question

If we have different models (e.g. different M, λ etc.) to describe the data which should we choose?

- If we have enough data then evaluate every model (fully trained on the training set) on the validation set. Choose the one with lowest validation test error.
- If data is scarce one can use S-fold cross validation (see next slide).
- One could use information criteria, which penalize complexity:
  - Akaike IC (AIC): Choose model with minimal:

$$M - \ln p(D|w_{\mathrm{ML}}).$$

  - Bayesian IC (BIC): Bayesian + crude approximations.
  - Full Bayesian: Penalties arise automatically.

Cross Validation

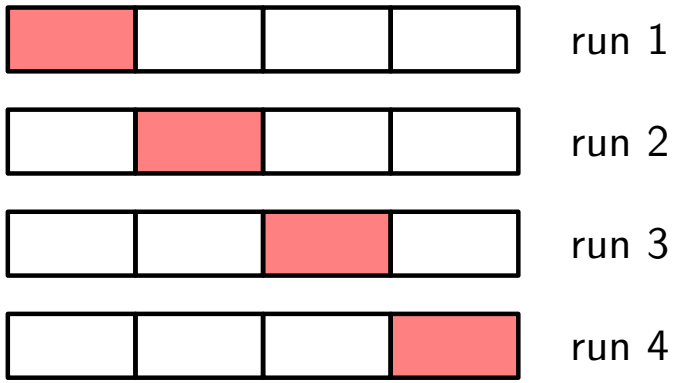

Figure: S-fold cross validation. (Bishop Table 1.18)

- Partition data into S groups (of similar size). In case data is very scarce use $S = N$ (leave-one-out cross validaton).
- Train model on $S - 1$ groups and evaluate error on the last.
- Repeat S-times by changing the left out group and avarage the validation errors to get a test error estimate.
- Do this for every model and choose the one with lowest such test error.

Expected Test Error: Preliminaries: Conditional Expectation

- Let X, ϵ be independent random variables with $\mathbb{E}[\epsilon] = 0$ and $T = h(X) + \epsilon$. Then:
- $\mathbb{E}[T|X] = \mathbb{E}[h(X)|X] + \mathbb{E}[\epsilon|X] = h(X)$. And furthermore:
- $$\begin{aligned}
\mathbb{E}[(T - g(X))^2] &= \mathbb{E}[(T - h(X) + h(X) - g(X))^2] \\
&= \mathbb{E}[(T - h(X))^2 + (h(X) - g(X))^2 \\
&\quad +2(T - h(X))(h(X) - g(X))] \\
&= \mathbb{E}[(T - h(X))^2] + \mathbb{E}[(h(X) - g(X))^2] \\
&\quad +2\mathbb{E}\mathbb{E}[[(T - h(X))(h(X) - g(X))|X]] \\
&= \mathbb{E}[(T - h(X))^2] + \mathbb{E}[(h(X) - g(X))^2] \\
&\quad +2\mathbb{E}[(\mathbb{E}[T|X] - h(X))(h(X) - g(X))] \\
&= \mathbb{E}[(T - h(X))^2] + \mathbb{E}[(h(X) - g(X))^2] \\
&\geqslant \mathbb{E}[(T - h(X))^2]
\end{aligned}$$
- I.e. under all functions $g(X)$ the function $h(X) = \mathbb{E}[T|X]$ minimized the quadratic distance $\mathbb{E}[(T - g(X))^2]$.

Expected Test Error: Bias - Variance - Decomposition

- Let X, ϵ be independent random variables with $\mathbb{E}[\epsilon] = 0$ and $T = h(X) + \epsilon$ and $D = (X_1, \ldots, X_N)$ i.i.d. instances of X and W a noisy parameter "learned" from D and y the predictive function. Then the expected (quadratic) test error is:
-
$$\begin{aligned} & \mathbb{E}[(T - y(X, W))^2] & \\ = \; & \mathbb{E}[(T - h(X))^2] + \mathbb{E}[(h(X) - y(X, W))^2] & \\ = \; & \mathbb{E}[(T - h(X))^2] & (\text{noise})^2 \\ & + \mathbb{E}[(h(X) - \mathbb{E}_D[y(X, W)])^2] & (\text{bias})^2 \\ & + \mathbb{E}[(\mathbb{E}_D[y(X, W)] - y(X, W))^2] & (\text{variance}) \end{aligned}$$

- Expected Test Error = Bias^2 + Variance + Noise^2,
- Bias: measures the "difference" between desired regression function h and the avarage prediction over all data sets.
- Variance: measures sensitivity of y to particular choice of data set around the average over all data sets.
- Noise: just a constant coming from the variance of ϵ.

Bias - Variance - Estimator

- Take L (e.g. $L = 100$) groups, each of which has N (e.g. $N = 25$) independent observations $D^{(i)} = (x_{i,1}, \ldots, x_{i,N})$, $i = 1, \ldots, L$.
- For every $i = 1, \ldots, L$ fit a model $y^{(i)}$ on $D^{(i)} = (x_{i,1}, \ldots, x_{i,N})$.
- Average prediction estimate: $\overline{y}(x) = \frac{1}{L} \sum_{i=1}^{L} y^{(i)}(x)$.
- Take test data $D' = (x_1, \ldots, x_{N'})$ and calculate the estimates:
- $\text{Bias}^2 = \frac{1}{N'} \sum_{k=1}^{N'} (\overline{y}(x_k) - h(x_k))^2$,
- Variance$= \frac{1}{N'} \sum_{k=1}^{N'} \frac{1}{L} \sum_{i=1}^{L} (y^{(i)}(x_k) - \overline{y}(x_k))^2$.

Example: Bias - Variance with Regularization

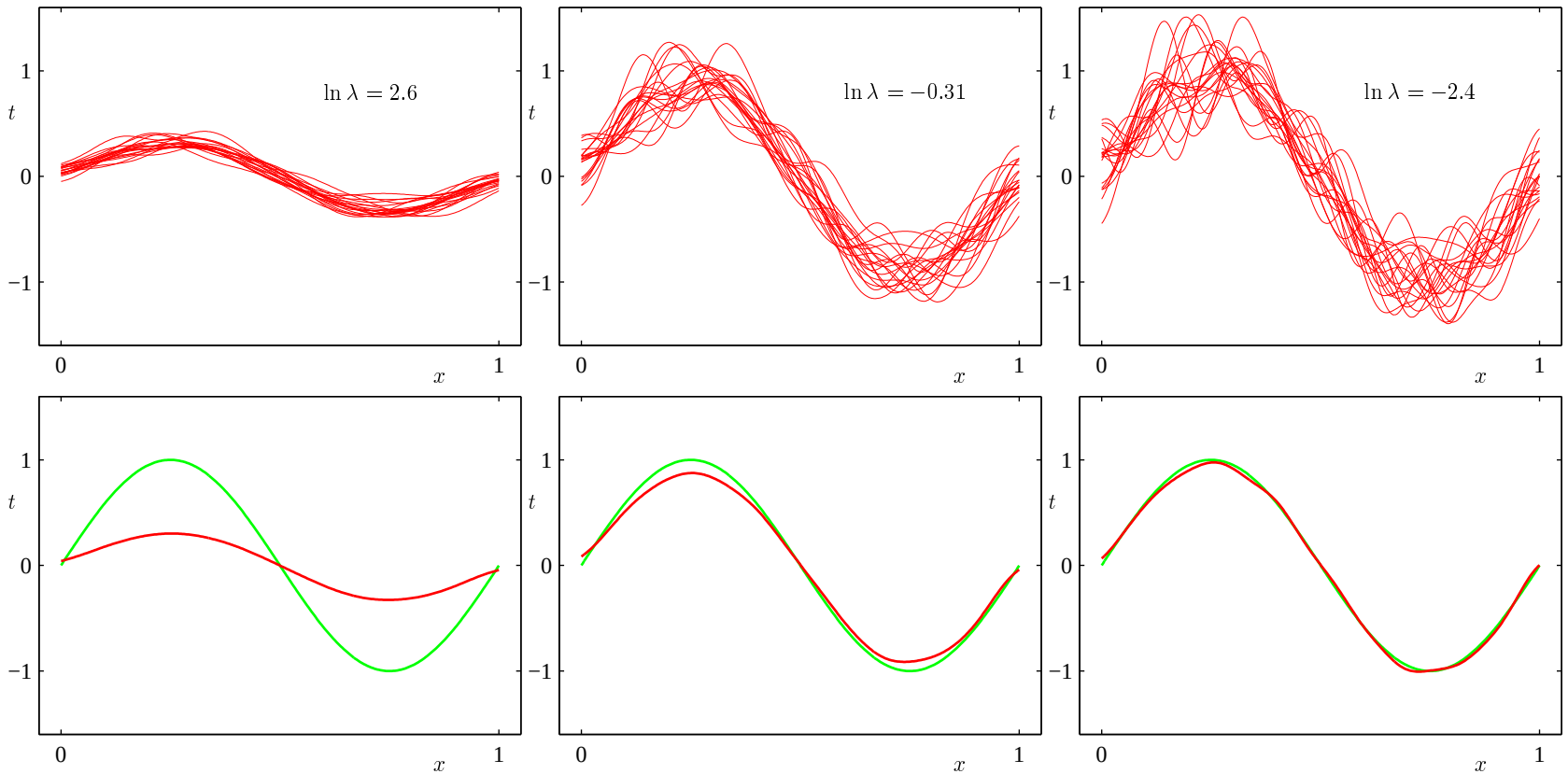

Figure: $L = 100$ data sets, each having $N = 25$ data points, $M - 1 = 24$ Gaussian basis functions. Top: 20 of the 100 fits by regularization parameter λ. Bottom: Avarage over all 100 fits (red) along with data generating function (green). Left: Low variance, high bias. Right: High variance, low bias. (Bishop 3.5)

Example: Bias - Variance - Decomposition

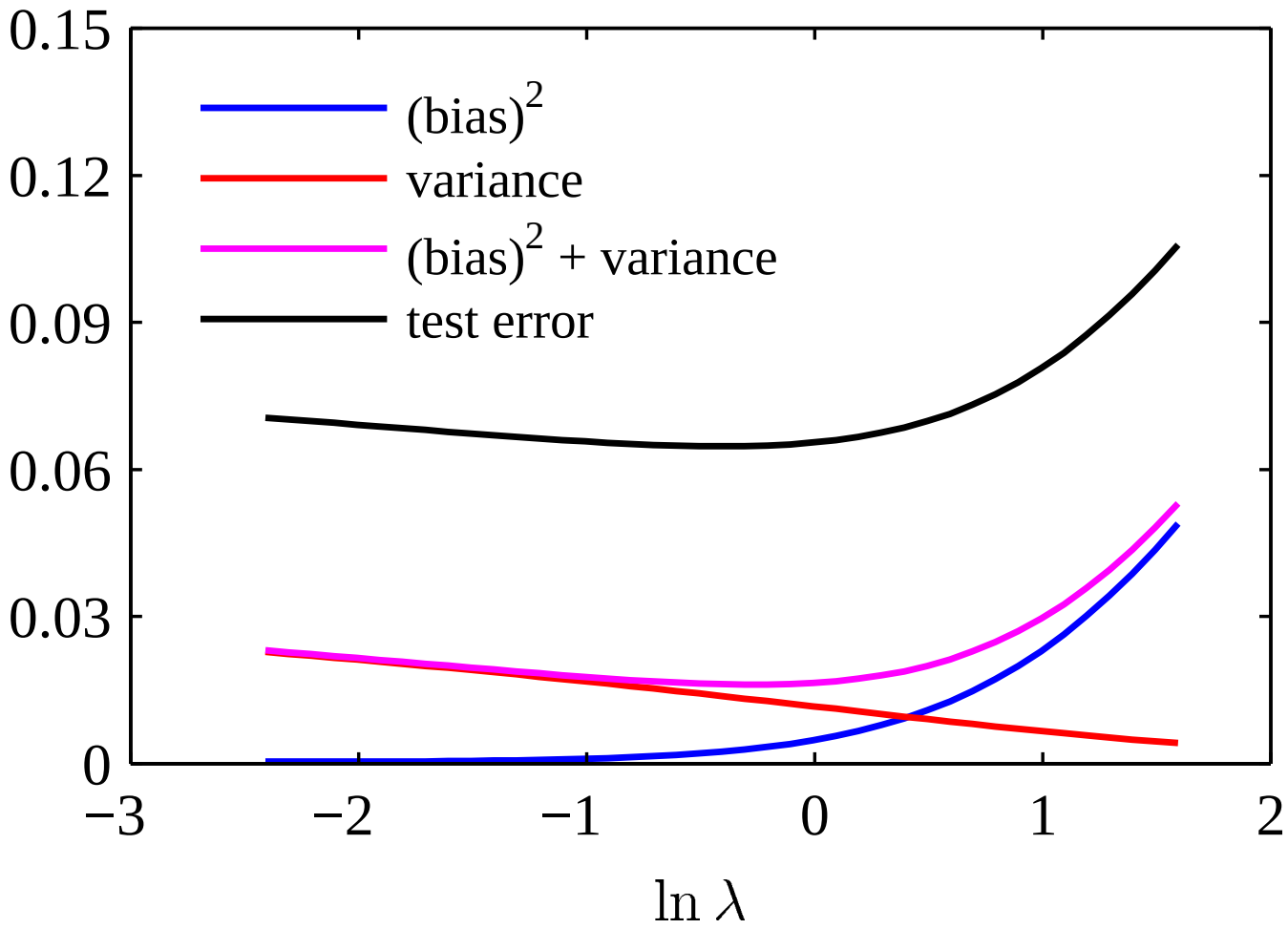

Figure: Estimated bias^2 and variance with $N' = 1000$ test data point. Minimal sum of bias^2+variance for regularization parameter $\ln \lambda = -0.31$. (Bishop 3.6)

Bayesian Linear Regression (I)

- Training data: $D = (x_1, \ldots, x_N)^T$ with targets $T = (t_1, \ldots, t_N)^T$.
- Linear Basis Function Model: $t = w^T \phi(x) + \epsilon$ with Gaussian noise ϵ and parameters $w = (w_0, \ldots, w_{M-1})^T \in \mathbb{R}^M$.
- So for (x, t) we have $p(t|x, w) = \mathcal{N}(t|w^T \phi(x), \beta^{-1})$, where $\beta = 1/\sigma^2$ is the precision, leading to:
- Likelihood: $p(T|w, D, \beta) = \prod_{i=1}^{N} \mathcal{N}(t_i|w^T \phi(x_i), \beta^{-1})$.
- Bayesian Approach: Gaussian Prior: $p(w) = \mathcal{N}(w|\mu_0, \Sigma_0)$ with mean μ_0 and covariance Σ_0. This leads to:
- Gaussian Posterior: $p(w|T, D, \beta) = \mathcal{N}(w|\mu_N, \Sigma_N)$ with mean μ_N and covariance Σ_N calculated to (see Matrix Cook Book):

$$\begin{aligned} \Sigma_N &= (\Sigma_0^{-1} + \beta \Phi^T \Phi)^{-1} \\ \mu_N &= \Sigma_N(\Sigma_0^{-1}\mu_0 + \beta \Phi^T T) \end{aligned}$$

- So the Maximum A Posteriori estimate is $w_{\mathrm{MAP}} = \mu_N$.

Bayesian Linear Regression (II)

- Special case: $\mu_0 = 0$ and $\Sigma_0 = \alpha^{-1}\mathbb{1}$ with $\alpha > 0$. Leading to:
- Gaussian Prior: $p(w|\alpha) = \mathcal{N}(w|0, \alpha^{-1}\mathbb{1})$.
- Gaussian Posterior: $p(w|T, D, \alpha, \beta) = \mathcal{N}(w|\mu_N, \Sigma_N)$ with:

$$\begin{aligned} \Sigma_N &= (\alpha\mathbb{1} + \beta\Phi^T\Phi)^{-1} \\ \mu_N &= \beta\Sigma_N\Phi^T T. \end{aligned}$$

- Maximizing the log-posterior (with Gaussian prior) w.r.t. w:

$$\begin{aligned} \ln p(w|T, D, \alpha, \beta) &= -\frac{\beta}{2}\sum_{i=1}^{N}(t_i - w^T\phi(x_i))^2 - \frac{\alpha}{2}w^T w + \text{const} \\ &= -\beta\left(E(D, T, w) + \frac{\alpha}{2\beta}||w||_2^2\right) + \text{const.} \end{aligned}$$

is equivalent to Ridge Regression with regularization parameter $\lambda = \frac{\alpha}{\beta}$.

Sequential Bayesian Learning for Linear Regression

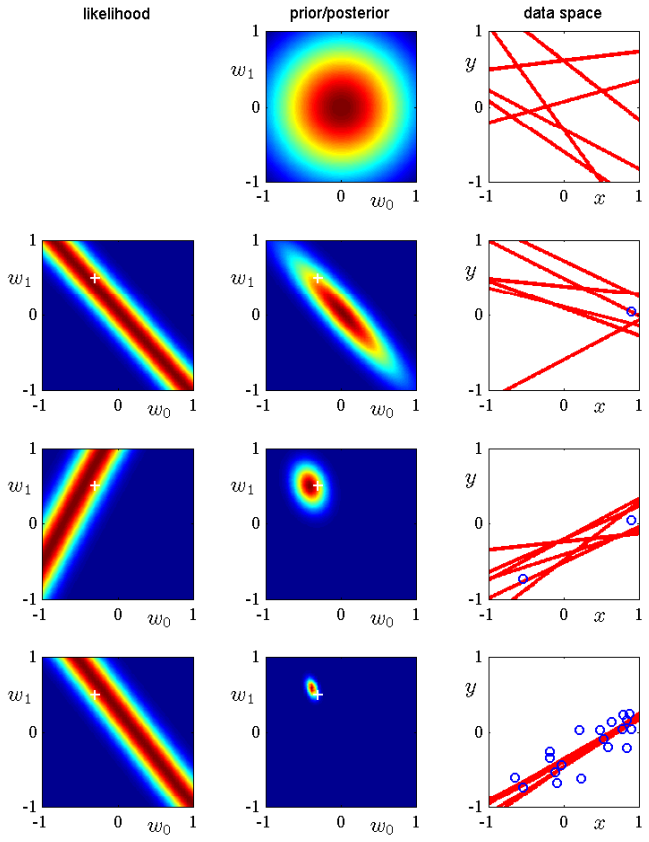

Figure: Predictive functions $y(x, w) = w_0 + w_1 \cdot x$. Likelihood $p(t|x, w)$, prior/posterior $p(w|D)$. White cross = true value (Bishop 3.7)