Machine Learning 1

Lecture 02 - Statistics - Supervised Learning: Linear Regression

Patrick Forré

The Rules of Probability Theory

Theorem (The Rules of Probability Theory)

For random variables $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$ we have the following rules:

	discrete RV	*continuous RV*
σ-Additivity	$\mathbb{P}(X \in A) = \sum_{x \in A} p(x)$	$\mathbb{P}(X \in A) = \int_A p(x)dx$
Positivity	$p(x) \geqslant 0$	$p(x) \geqslant 0$
Normalization	$\sum_{x \in \mathcal{X}} p(x) = 1$	$\int_{\mathcal{X}} p(x)dx = 1$
Sum Rule	$p(x) = \sum_{y \in \mathcal{Y}} p(x, y)$	$p(x) = \int_{\mathcal{Y}} p(x, y)dy$
Product Rule	$p(x, y) = p(x\|y) \cdot p(y)$	$p(x, y) = p(x\|y) \cdot p(y)$

Bayes' Rule

Theorem (Bayes)

For random variables $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$ we have the following rule:

$$p(y|x) = \frac{p(x|y) \cdot p(y)}{p(x)} = \begin{cases} \frac{p(x|y) \cdot p(y)}{\sum_{y' \in \mathcal{Y}} p(x|y') \cdot p(y')} & \text{for discrete RV} \\ \frac{p(x|y) \cdot p(y)}{\int_{\mathcal{Y}} p(x|y') \cdot p(y') dy'} & \text{for continuous RV} \end{cases}$$

I.a.w. the conditioning can be "exchanged" by this rule.

Maximum Likelihood Estimation

- Data set $D = (x_1, \ldots, x_N)$ of N independent observations given.
- We are presented with a class of probability distributions $\{p(x|w)|w \in \mathcal{W}\}$ for x, where $\mathcal{W}$ is an index set (in some $\mathbb{R}^d$).
- Goal: Find an index w^* such that $p(x|w^*)$ "best" explains the occurence of the data D,
- i.e. such that D appears to be a realization of i.i.d. (independent and identically distributed) random variables $X_1, \ldots, X_N$ each of which is distributed like $p(x|w^*)$.
- Maximum Likelihood Principle: The most likely "explanation" of D is given by the index w which maximizes the likelihood $p(D|w)$ (joint distribution).

Maximum Likelihood Estimation (II)

- The Maximum Likelihood Estimator w_{ML} is determined by:

$$\begin{aligned} w_{\mathrm{ML}} &:= \mathrm{argmax}_{w \in \mathcal{W}}\, p(D|w) \\ &= \mathrm{argmax}_{w \in \mathcal{W}} \prod_{i=1}^{N} p(x_i|w) \\ &= \mathrm{argmax}_{w \in \mathcal{W}} \sum_{i=1}^{N} \log p(x_i|w) \\ &= \mathrm{argmin}_{w \in \mathcal{W}} \left\{ -\sum_{i=1}^{N} \log p(x_i|w) \right\} \end{aligned}$$

- Putting $E(x_i; w) := -\log p(x_i|w)$ leaves us with minimizing the error function $E(D; w) := \sum_{i=1}^{N} E(x_i; w)$ w.r.t. w.
- In case $\log p(D|w)$ is differentiable w.r.t. w we can take the derivative w.r.t. w, set it to zero and solve for w to get w_{ML}.
- For predicting the value of new data x' we now use the distribution $p(x'|w_{\mathrm{ML}})$.

Example: Maximum Likelihood Estimator for Gaussian

- We have $D = (x_1, \ldots, x_N)$ and $p(x|w) = \mathcal{N}(x|\mu, \sigma^2)$ with $w = (\mu, \sigma^2)$. So:

$$p(D|\mu, \sigma^2) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^N \prod_{i=1}^{N} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right).$$

- Taking log, putting derivatives to zero and solving for μ and σ^2 we get:

$$\mu_{\mathrm{ML}} = \frac{1}{N}\sum_{i=1}^{N} x_i \quad \text{and} \quad \sigma^2_{\mathrm{ML}} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_{\mathrm{ML}})^2.$$

- We get: $\mathbb{E}[\mu_{\mathrm{ML}}] = \mu$ and $\mathbb{E}[\sigma^2_{\mathrm{ML}}] = \frac{N-1}{N}\sigma^2$. So the variance is estimated to low, i.e. we have a Bias.
- $\tilde{\sigma}^2 := \frac{N}{N-1}\sigma^2_{\mathrm{ML}}$ is an unbiased estimator for σ^2.

Bayesian Prediction

- Data set $D = (x_1, \ldots, x_N)$ of N independent observations given.
- We are presented with a class of probability distributions $\{p(x|w)|w \in \mathcal{W}\}$ for x, where $\mathcal{W}$ is an index set (in some $\mathbb{R}^d$).
- Goal: Estimate the distribution of a new data point x'.
- Bayesian Principle: Instead of searching for one optimal w consider all $w \in \mathcal{W}$ simultaneously and assign a probability distribution $p(w)$ over it, reflecting the plausability of each w.
- $p(w)$ is called the prior distribution of w.
- Then adjust/update $p(w)$ with the occurence of data D to $p(w|D)$, making some w more plausible and others less (in accordance with D).
- $p(w|D)$ is called the posterior distribution of w after observing D.

Bayesian Prediction (II)

- The posterior $p(w|D)$ can be computed with Bayes' Rule:

$$p(w|D) = \frac{p(D|w)}{p(D)} \cdot p(w).$$

- $p(D|w)$ is called the likelihood and $p(D)$ the evidence.
- Before learning D the predictive distribution is:

$$p(x') = \int_S p(x'|w)p(w)dw.$$

- After learning D the predictive distribution becomes:

$$p(x'|D) = \int_{\mathcal{W}} p(x'|D,w)p(w|D)dw = \int_{\mathcal{W}} p(x'|w)p(w|D)dw.$$

Maximum A Posteriori Probability Estimation

- Data $D = (x_1, \ldots, x_N)$ of N independent observations given.
- Probability distributions $\{p(x|w)|w \in \mathcal{W}\}$ for x given.
- Bayesian setting: probability distribution $p(w)$ given.
- Maximum a Posteriori Principle: The most likely "explanation" of D is not just given by the index w which maximizes $p(D|w)$, but which maximizes the a posteriori $p(w|D) = \frac{p(D|w)p(w)}{p(D)}$.
- The Maximum A Posteriori Estimator w_{MAP} is:

$$\begin{aligned} w_{\text{MAP}} &:= \text{argmax}_{w \in \mathcal{W}}\, p(w|D) \\ &= \text{argmax}_{w \in \mathcal{W}}\, p(D|w)p(w) \\ &= \text{argmax}_{w \in \mathcal{W}}\, \log p(D|w) + \log p(w) \\ &= \text{argmax}_{w \in \mathcal{W}}\, \sum_{i=1}^{N} \log p(x_i|w) + \log p(w). \end{aligned}$$

Example: Maximum A Posteriori Estimator for Gaussian

- We have $D = (x_1, \ldots, x_N)$ and assume $p(x|\mu) = \mathcal{N}(x|\mu, \sigma^2)$, where σ^2 is assumed to be fixed.
- We assume the prior $p(\mu) = \mathcal{N}(\mu|\mu_0, \tau^2)$. So:

$$p(D|\mu) \cdot p(\mu) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^N \prod_{i=1}^N \exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right) \cdot \frac{1}{\sqrt{2\pi\tau^2}} \exp\left(-\frac{(\mu-\mu_0)^2}{2\tau^2}\right).$$

- Taking log, putting derivatives to zero, we get:

$$\mu_{\text{MAP}} = \frac{N\tau^2}{N\tau^2 + \sigma^2}\left(\frac{1}{N}\sum_{i=1}^N x_i\right) + \frac{\sigma^2}{N\tau^2 + \sigma^2}\mu_0.$$

- So μ_{MAP} is a convex linear combination between the sample mean μ_{ML} and the prior mean μ_0.

Supervised Learning: General Concept

- Goal: Predict target variable t from corresponding data x.
- Training Data: We have a data set $D = (x_1, \ldots, x_N)$ of N observations together with their target variables (estimates) $T = (t_1, \ldots, t_N)$ given ($\rightarrow$ supervision).
- Model: We choose a class of functions in x: $\{y(x, w) | w \in \mathcal{W}\}$ as possible prediction function (with the aim $y(x, w) \approx t$).
- Error/Loss functions: For every data set $D' = (x'_1, \ldots, x'_L)$ with targets $T' = (t'_1, \ldots, t'_L)$ we can measure the "loss" or "error" between the predictions $y(x'_i, w)$ and the target t'_i by chosen functions: $E(y(D', w), T')$.
- Strategy: Minimize the error $E(y(x, w), t)$ w.r.t. w for all known and unknown data points (x, t).
- Caution: We not only want the training error $E(y(D, w), T)$ to be low, but more importantly the generalization/test error $E(y(x, w), t)$ to be minimal for unknown cases (x, t).

Supervised Learning: Evaluating the Error in Praxis

Question

How do we measure the error of a prediction for unknown data sets (x, t) (since we do not know the correct target variable t)?

Hold out known data! Given that our known data set $D = (x_1, \ldots, x_N)$ with targets $T = (t_1, \ldots, t_N)$ is big enough then we randomly divide the data set D into three groups:

1. training set ($D_{\text{tr}} \approx 60\%$ of D): Only D_{tr} and T_{tr} will be used for training (i.e. finding the "right" w^* for $y(x, w^*)$).
2. validation set ($D_{\text{val}} \approx 20\%$ of D): D_{val} and T_{val} will be used for monitoring the estimated test error: $E(y(D_{\text{val}}, w^*), T_{\text{val}})$.
3. test set ($D_{\text{test}} \approx 20\%$ of D): D_{test} will only be allowed to be used once (!!!) for reporting the estimated test error: $E(y(D_{\text{test}}, w^*), T_{\text{test}})$.

If D is not big enough, we rely on weaker evaluation techniques.

Supervised Learning: How to train? - The Naive Approach

Strategy (1. Naive Approach: Minimize the training error)

Use $y(x, w^*)$ as prediction function, where w^* minimizes the training error $E(y(D_{\text{tr}}, w), T_{\text{tr}})$ w.r.t. w.

Caution!

We will see in the next examples that this strategy preferes overly complex models, which perform bad on test sets.

Strategy (2. Regularized, Bayesian, Shrinkage Approach)

Penalizing complexity during training. See later.....

Linear Basis Function Model: Basis Functions

- The training data is $D_{\text{tr}} = (x_1, \ldots, x_N)$, 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 $M-1$ basis functions / "features" of x:

$$\phi_1, \ldots, \phi_{M-1}.$$

- We will then consider the model of the following form:

$$y(x, w) = w_0 + \sum_{i=0}^{M-1} w_i \cdot \phi_i(x),$$

with $w = (w_0, \ldots, w_{M-1}) \in \mathbb{R}^M$ $(= \mathcal{W})$.

- These functions are linear in w!
- Putting $\phi_0(x) = 1$ and $\phi(x) = (\phi_0(x), \ldots, \phi_{M-1}(x))^T$ we can write:

$$y(x, w) = w^T \phi(x).$$

Example: Basis Functions (I)

- $\phi_i(x) = x_{,i}$, the projection onto the i-th component, where $x = (x_{,1}, \ldots, x_{,D})$. Then:

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

  are linear functions in w and x (D-dimensional Linear Regression).
- $\phi_i(x) = x^i$, the i-power map. Then:

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

  (1-dimensional Polynomial Regression).

Example: Basis Functions (II)

- $\phi_i(x) = \exp\left(-\frac{1}{2}(x-\mu_i)^T\Sigma^{-1}(x-\mu_i)\right)$,
  Gaussian basis functions with fixed μ_i and Σ. Then:

$$y(x,w) = w_0 + \sum_{i=1}^{M-1} w_i \cdot \exp\left(-\frac{1}{2}(x-\mu_i)^T\Sigma^{-1}(x-\mu_i)\right).$$

- $\phi_i(x) = \sigma\left(\frac{x-\mu_i}{s}\right)$, with fixed μ_i, s and where $\sigma(a) = \frac{1}{1+\exp(-a)}$ is the logistic sigmoid function. Then:

$$y(x,w) = w_0 + \sum_{i=1}^{M-1} w_i \cdot \sigma\left(\frac{x-\mu_i}{s}\right).$$

Example: Basis Functions (III)

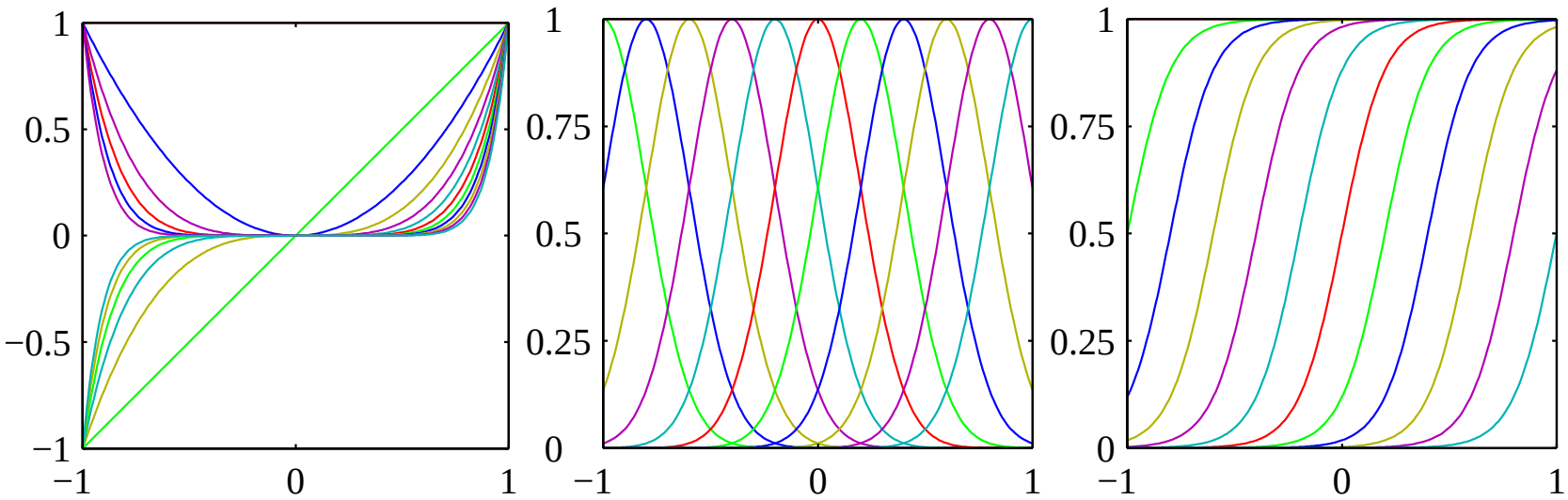

Figure: Polynomials, Gaussian, Sigmoids (Bishop 3.1)

Linear Basis Function Model: Sum-of-Squares Error

- For all Linear Basis Function Models we will use a quadratic error function:

$$E(x, t, w) := \frac{1}{2}(t - y(x, w))^2$$

- For date sets $D' = (x'_1, \ldots, x'_L)$ with targets $T' = (t'_1, \ldots, t'_L)$ this leads to the sum-of-squares error function:

$$E(D', T', w) := \sum_{i=1}^{L} E(x'_i, t'_i, w) = \frac{1}{2} \sum_{i=1}^{L} (t'_i - y(x'_i, w))^2.$$

- For comparison of errors of different sample size L one also uses the root-mean-squared error:

$$E_{\mathrm{RMS}}(D', T', w) := \sqrt{\frac{1}{L} \sum_{i=1}^{L} (t'_i - y(x'_i, w))^2}.$$

Example: Error Function

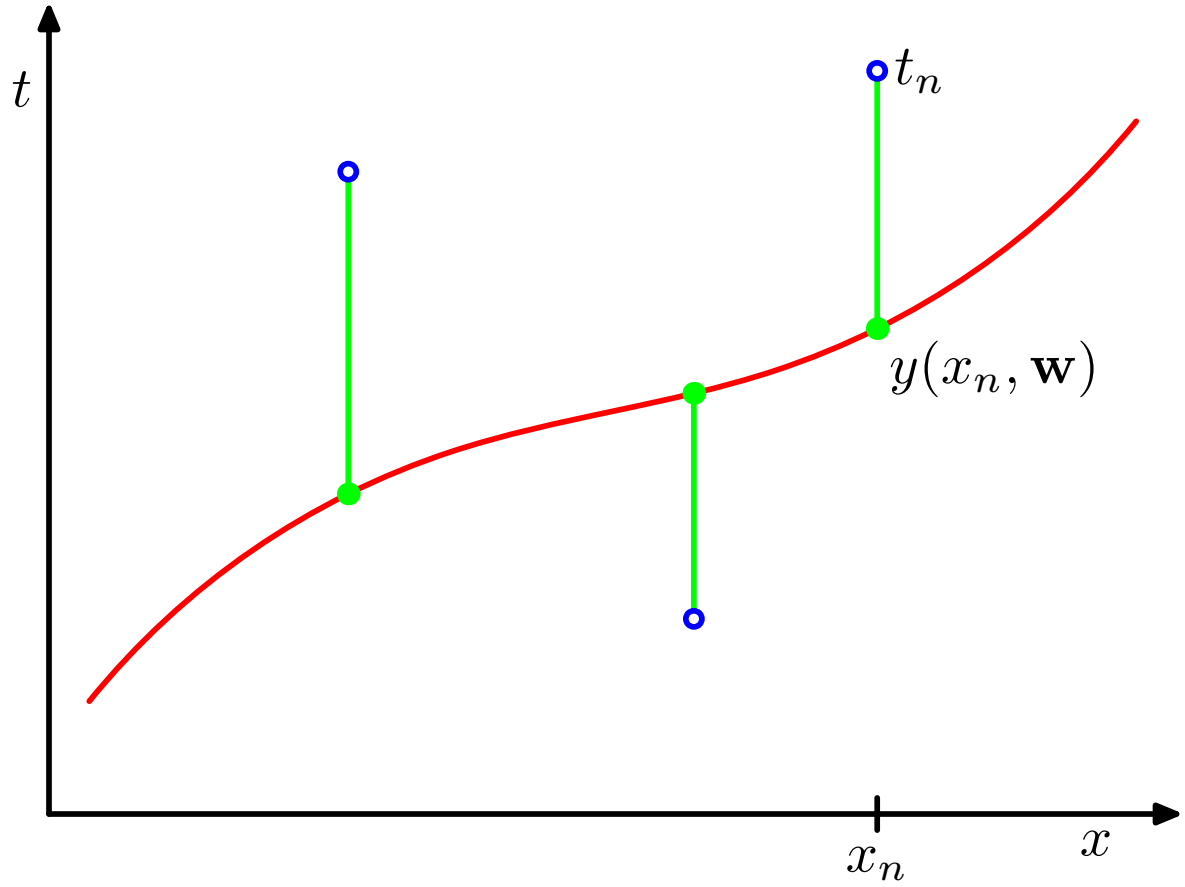

Figure: Errors are given by half the square of the green bars (Bishop 1.3)

Linear Basis Function Model: Least-Squares-Solution

Theorem (Closed form of the least-squares-solution)

- *Training data* $D = (x_1, \ldots, x_N)^T$, *targets* $T = (t_1, \ldots, t_N)^T$, *the linear basis model* $\{y(x, w) = w^T \phi(x) | w \in \mathbb{R}^M\}$ *and error function* $E(D, T, w) = \frac{1}{2} \sum_{i=1}^{N} (t_i - y(x_i, w))^2$ *given.*
- *Put:*

$$\Phi := \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_{M-1}(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \cdots & \phi_{M-1}(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x_N) & \phi_1(x_N) & \cdots & \phi_{M-1}(x_N) \end{pmatrix}$$

- *Then* $w^* = (\Phi^T \Phi)^{-1} \Phi^T T$ *is the unique minimizer of* $E(D, T, w)$ *w.r.t.* w *(in case the matrix inverse exists).*

Linear Basis Function Model: Maximum Likelihood Solution

- Assume that the target t is given by the deterministic function $y(x, w) = w^T \phi(x)$ overlayed with random noise ϵ:

$$t = y(x, w) + \epsilon, \qquad \epsilon \sim \mathcal{N}(e|0, \beta^{-1}),$$

where $\beta = 1/\sigma^2$ is the precision. Then:

- $p(t|x, w, \beta) = \mathcal{N}(t|y(x, w), \beta^{-1})$ and
- $p(T|D, w, \beta) = \prod_{i=1}^{N} \mathcal{N}(t_i|y(x_i, w), \beta^{-1})$ and thus:

$$\begin{aligned} w_{\mathrm{ML}} &:= \operatorname{argmax}_{w \in \mathcal{W}} p(T|D, w, \beta) \\ &= \operatorname{argmin}_{w \in \mathcal{W}} \left\{ -\sum_{i=1}^{N} \log(t_i|x_i, w, \beta) \right\} \\ &= \operatorname{argmin}_{w \in \mathcal{W}} \left(\frac{N}{2} \ln(\frac{2\pi}{\beta}) + \frac{\beta}{2} \sum_{i=1}^{N} (t_i - y(x_i, w))^2 \right) \\ &= \operatorname{argmin}_{w \in \mathcal{W}} E(D, T, w) \\ &= w* = (\Phi^T \Phi)^{-1} \Phi^T T. \end{aligned}$$

- So the Maximum Likelihood Principle for Gaussian noise gives the minimizer of the sum-of-squared error function.

Linear Basis Function Model: Maximum Likelihood Solution

- Special case:

$$w_{\mathrm{ML},0} = \overline{t} - \sum_{i=1}^{M-1} w_{\mathrm{ML},i}\overline{\phi_i},$$

where $\overline{t} = \frac{1}{N}\sum_{k=1}^{N} t_k$ and $\overline{\phi_i} = \frac{1}{N}\sum_{j=1}^{M-1} \phi_i(x_j)$.
So $w_{\mathrm{ML},0}$ compensated for the differences between the averages of the target values and of the basis function values.

- The Maximum Likelihood Estimator for the variance and precision $\beta = \frac{1}{\sigma^2}$ is:

$$\sigma^2_{\mathrm{ML}} = \frac{1}{\beta_{\mathrm{ML}}} = \frac{1}{N}\sum_{i=1}^{N}(t_i - w^T_{\mathrm{ML}}\phi(x_i))^2.$$

This is the residual variance of the target values around the regression function.

Geometry of the sum-of-squares error function

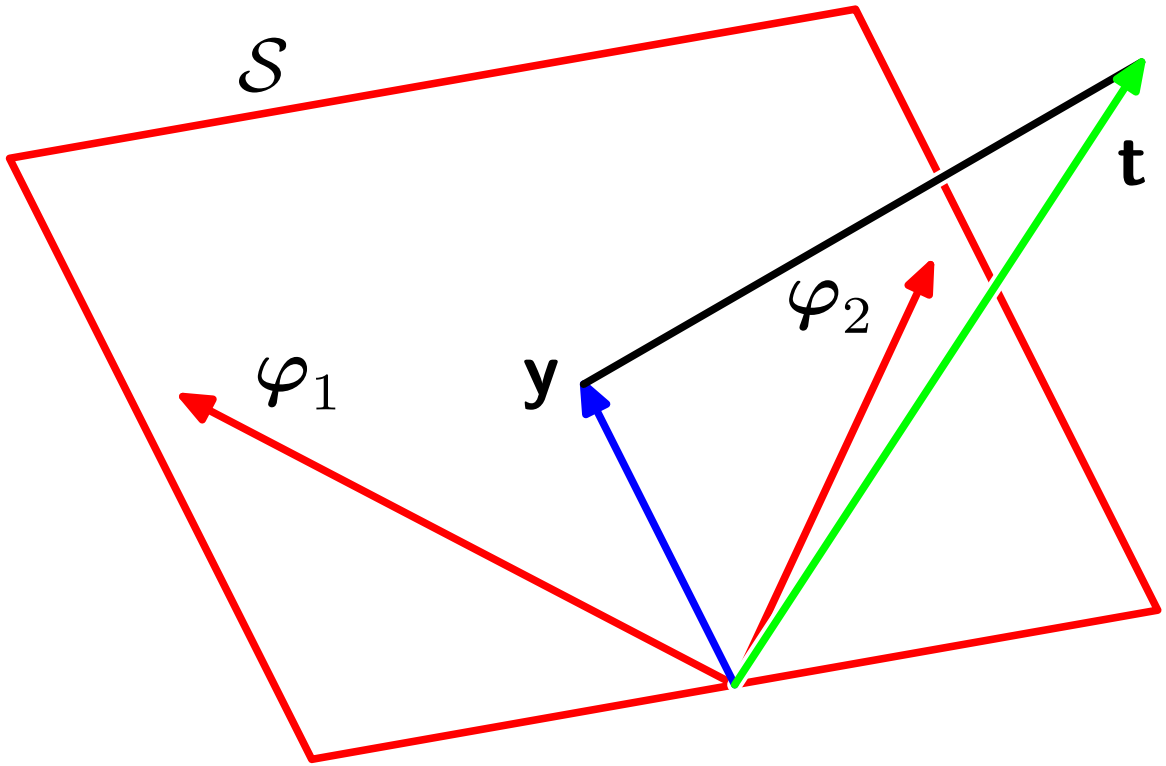

Figure: (Bishop 3.2) $\mathbf{t} = (t_1, \ldots, t_N)^T$, $\varphi_j = (\phi_j(x_1), \ldots, \phi_j(x_N))^T$, $\mathbf{y} = (y(x_1, w), \ldots, y(x_N, w))^T$. E is then (half) the square of the black line and $\mathbf{y}_{\mathrm{ML}}$ is the orthogonal projection of $\mathbf{t}$ into the plane generated by $\varphi_0, \ldots, \varphi_{M-1}$.

Sequential ("On-line") Learning: Stochastic Gradient Descent

- For big data sets $N >> 1$ computing the closed form solution $w_{\mathrm{ML}} = (\Phi^T \Phi)^{-1} \Phi^T T$ by matrix inversion is ineffecient and also needs all the data $x_1, \ldots, x_N$ at once.
- Instead we can use the least-mean-squares algorithm:
- Carefully choose a learning rate $\eta > 0$ and initialize the starting vector $w^{(0)}$, then iteratively use the "update rule":

$$\begin{aligned} w^{(n+1)} &:= w^{(n)} - \eta \cdot \nabla_w E(x_n, t_n, w) \\ &= w^{(n)} - \eta \cdot (t_n - w^{(n)T} \phi(x_n)) \phi(x_n) \end{aligned}$$

- Since $E(D, T, w) = \sum_{i=1}^{N} E(x_i, t_i, w)$ and E is convex in w the algorithm converges to w_{ML} if η is small enough.
- If η is too big the algorithm does not converge, if η is too small then the algorithm is too slow.

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.
- We generate the function $f(x) = \sin(2\pi x)$ 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 f 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

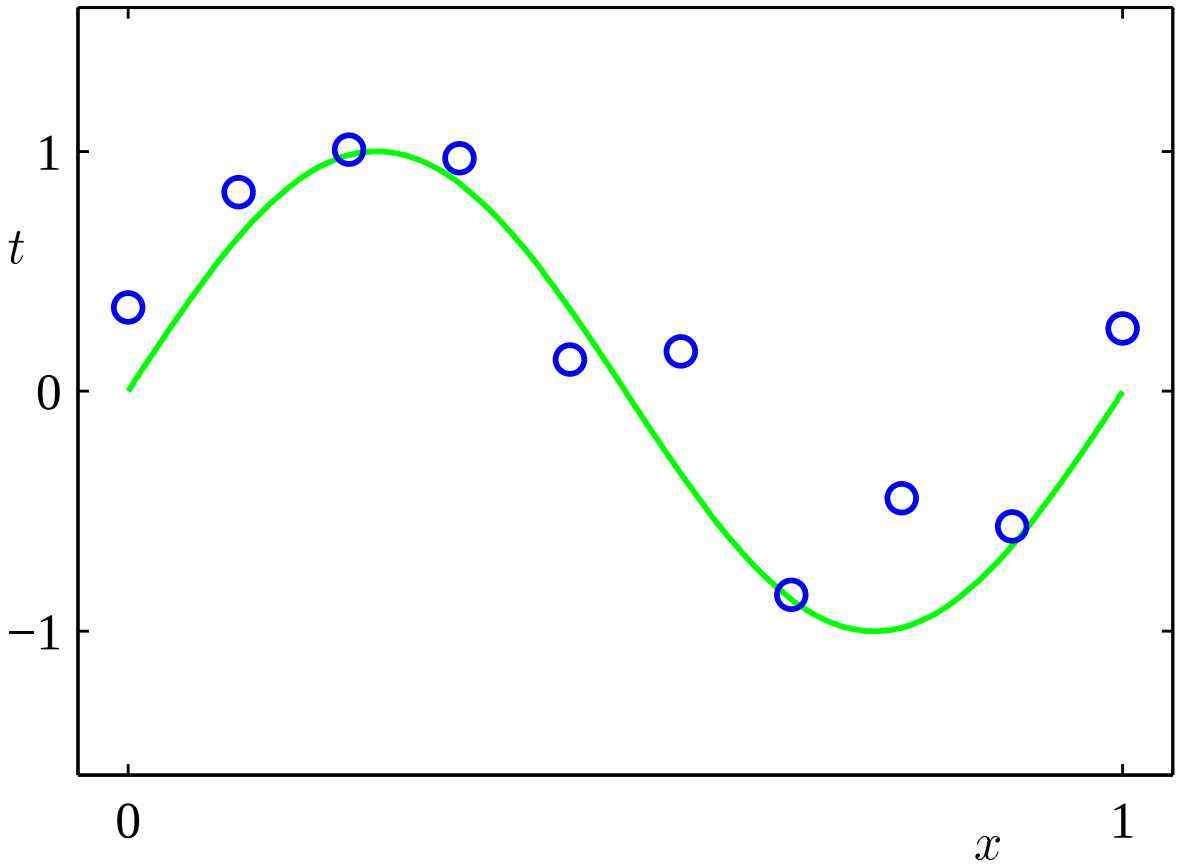

Figure: "Unknown" sinoidal curve ($f(x) = \sin(2\pi x)$) and observed data points ($t = f(x) + \epsilon$) (Bishop 1.2)

Example: Polynomial Regression

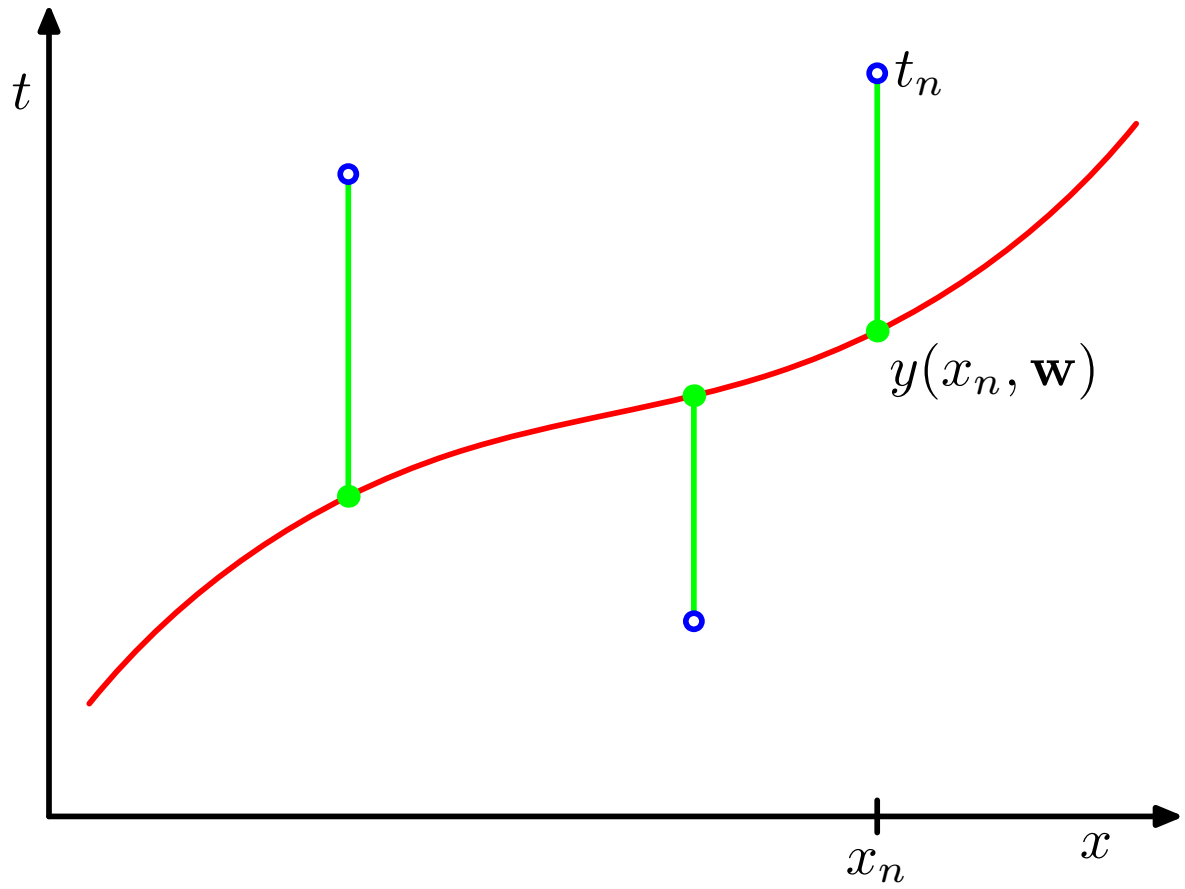

Figure: Varying the red line (polynomial function) to minimize the sum-of-squares of the green line (distance to data points) (Bishop 1.3)

Problems: Underfitting and Overfitting

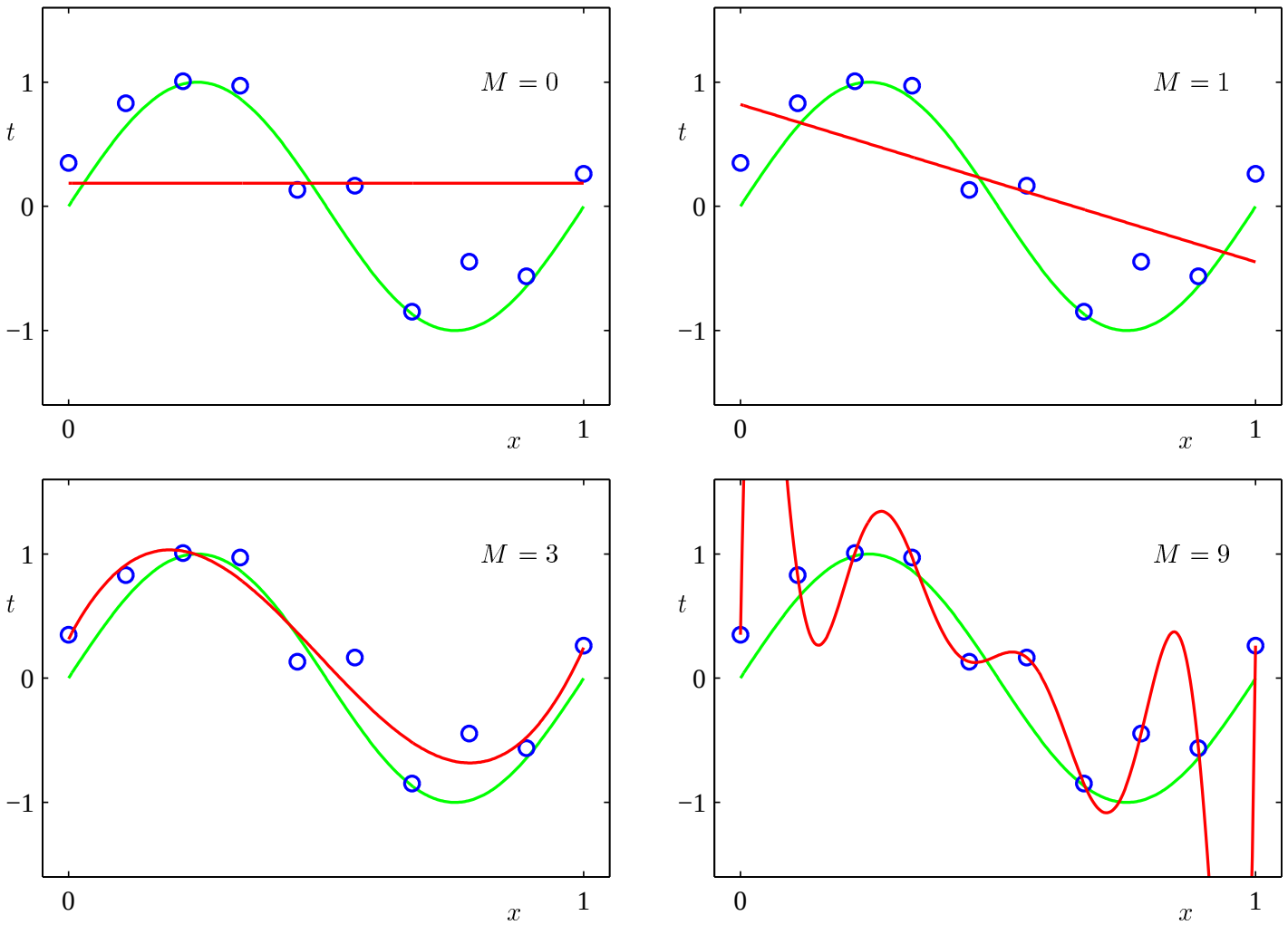

Table: Underfitting and Overfitting (Bishop 1.4)

Example: Training Error vs. Test Error by Complexity

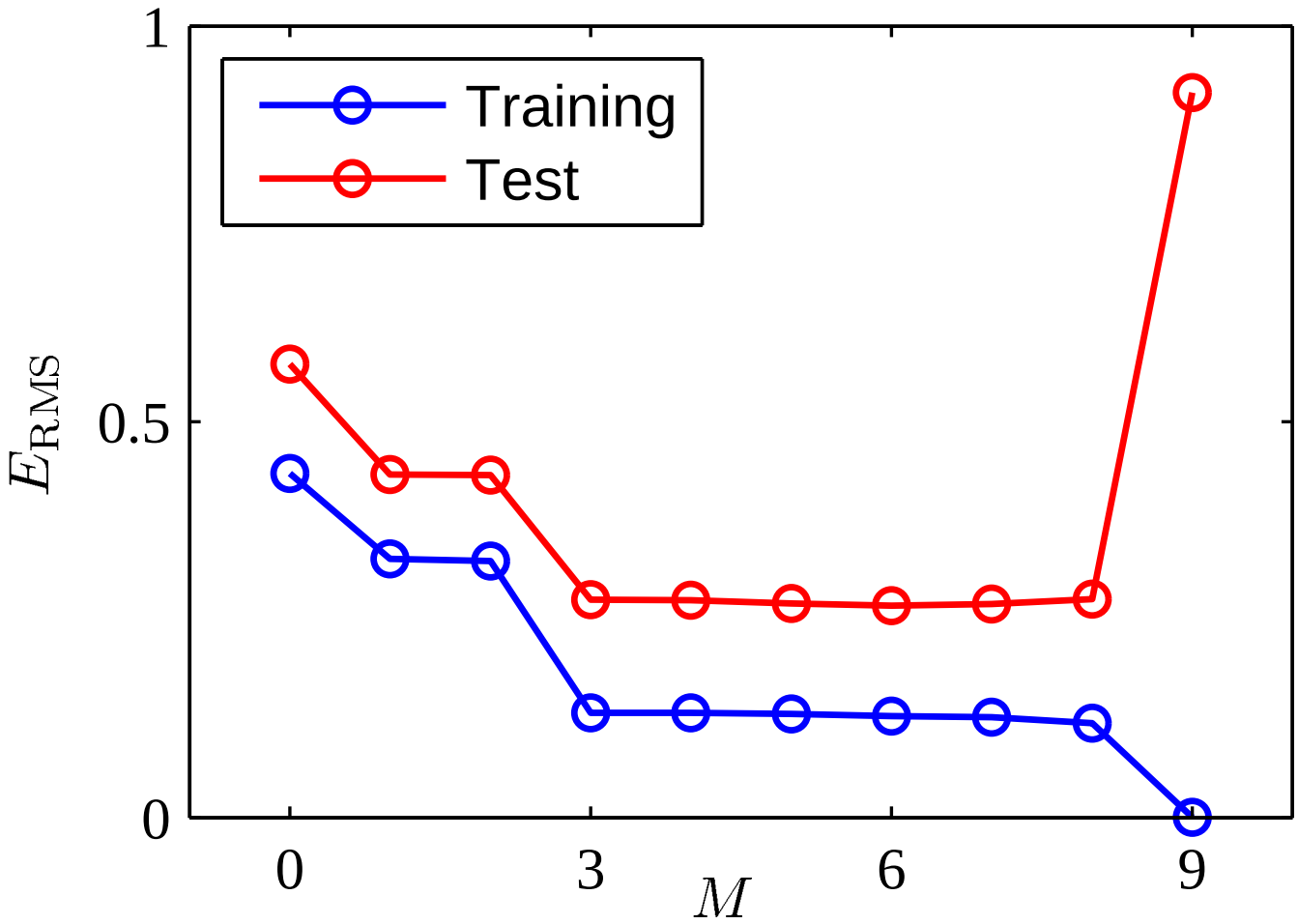

Figure: Training Error vs. Test Error by Complexity (Bishop 1.5)

Example: Size of Polynomial Coefficients

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)

Example: Sample Size vs. Overfitting

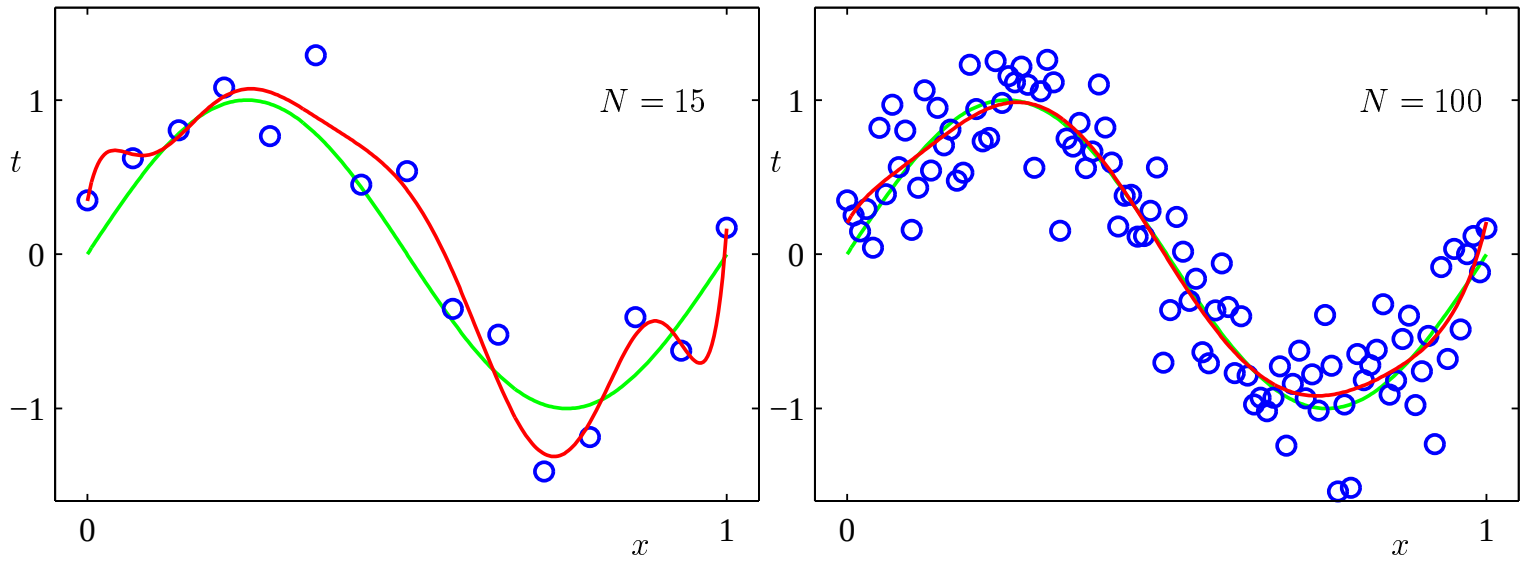

Figure: Polynomial fit of degree $M = 9$ and different number of data points N. Increase of N reduces overfitting (Bishop 1.6)