LUND UNIVERSITY

# L-2: Probability and Regression

Ted Kronvall, Cristian Sminchisescu

# Admin

## Teaching platform:

- We'll be using Moodle, available through a link at the course webpage http://www.ctr.maths.lu.se/course/machinlearn/2019/.
- Register to the course "Machine Learning 2019"
- Use enrollment key "clustering2019".

**By 8th of April at the latest!**

## Usage:

- Sign up for computer sessions!
- Find pdf prints of lecture slides.
- Fetch the assignment descriptions.
- Submit your assignment report and solutions
- Get grades and feedback.
- DO NOT USE MOODLE FOR MESSAGING!

LUND UNIVERSITY

# Admin

**Examination:**

- **Hand-in assignments**
  - Individual assessment
  - Approx. 2 weeks to submit report and solutions
  - Preferably solved using Matlab.
  - 70 % proficiency to pass
  - Two resubmissions allowed in total, provided a >= 30 % proficiency at initial submission.
  - No late submissions accepted!
- **Written exam**
  - Optional
  - Mandatory for higher grade (4-5).
  - Printed copies of course literature, lecture slides, and personal notes are ok to bring.

**By 10th of June at the latest!**

LUND UNIVERSITY

# Small recap from L-1

- Hypothesis space is the set of possible models considered in a machine learning algorithm - from where one model is selected by training.
- A dataset should typically be divided into three parts; estimation, validation, and testing.
- When comparing different models, it's their predictive capabilities, i.e., their generalization errors, that matter.
- The model and its capacity are typically selected using some hyperparameters.
- Cross-validation is a way to estimate the generalization error, where the estimate's uncertainty is taken into account.
- The no-free-lunch theorem is indeed a theorem and not only a conjecture.

LUND UNIVERSITY

# Today

- ~~Basic concepts; tasks, training, validation, regularization.~~
- Probability theory and optimization
- Linear regression
- Methods for classification

- Clustering methods
- Neural networks

- Convolutional and recurrent NNs

- Generative modeling

- Reinforcement learning
- Natural language processing
- Deep learning for computer vision

LUND UNIVERSITY

# Probability theory

LUND UNIVERSITY

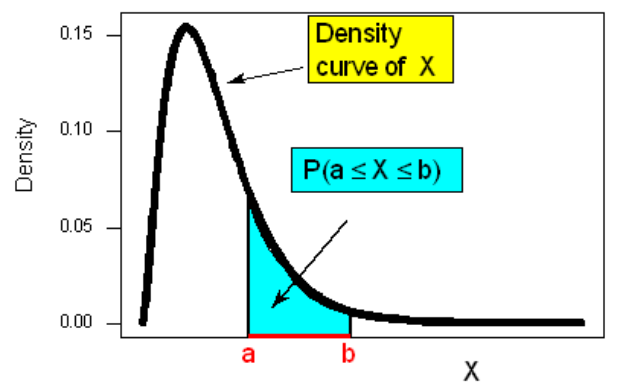
# Why probability?

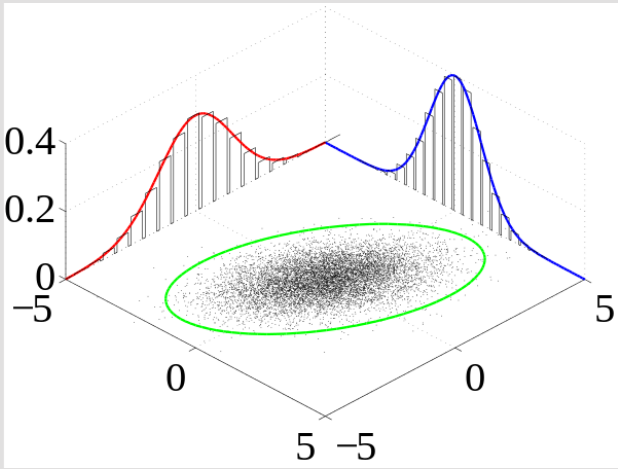
- **Uncertainty** is a key concept in pattern recognition and machine learning
- It arises both from *measurement noise* and from *finite size datasets*
- **Probability theory** provides consistent framework for the quantification and manipulation of uncertainty

LUND UNIVERSITY

# Random variables

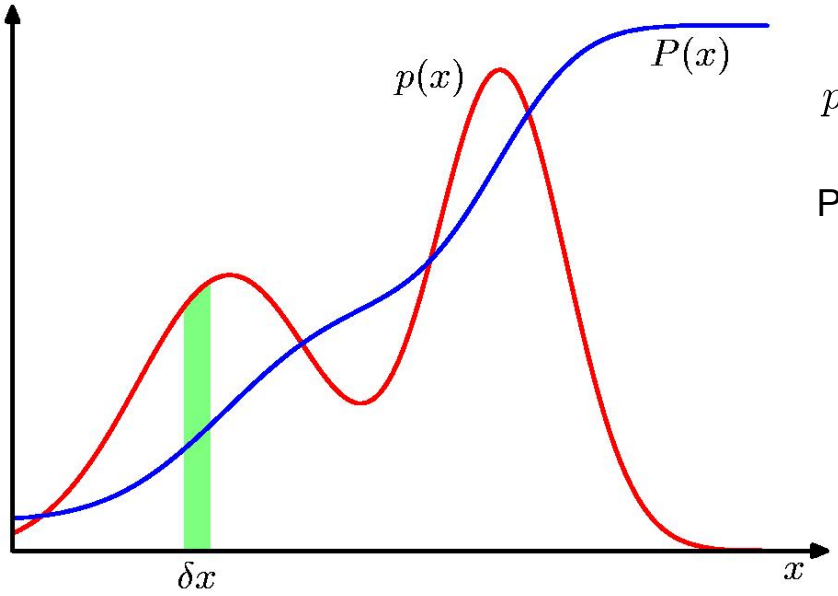
- A random variable $x$ takes values in certain regions according to certain probabilities, defined by its density function $p(x)$. The r.v. may be uni- or multivariate, and well as continuous or discrete.



# Random variables

## Probability Densities




$$p(x \in (a,b)) = \int_a^b p(x)\,\mathrm{d}x$$

Probability density, $\delta x \to 0$

$$P(z) = \int_{-\infty}^{z} p(x)\,\mathrm{d}x$$

Cumulative distribution function, $P'(x) = p(x)$

$$p(x) \geqslant 0 \qquad \int_{-\infty}^{\infty} p(x)\,\mathrm{d}x = 1$$

# Random variables

- Expectation:

$$E(x_1) = \int_{x_1} x_1 \cdot p(x_1) dx_1$$

- Variance:

$$V(x_1) = E(x_1^2) - E(x_1)^2$$

$$E(x_1^2) = \int_{x_1} x_1^2 \; p(x_1) dx_1$$

- Covariance:

$$C(x_1, x_2) = E(x_1 x_2) - E(x_1)E(x_2)$$

$$E(x_1 x_2) = \int_{x_1} \int_{x_2} x_1 x_2 \; p(x_1, x_2) dx_2 dx_1$$

# Random variables

## EXPECTATION

Expectation:

$$E(x_1) = \int_{x_1} x_1 \cdot p(x_1) dx$$

- The expectation is not a mean.
- The mean is a random variable.

Mean:

Given a dataset of independent and identically distributed variables, $x_1^{(n)}$, $n = 1, \ldots, N$, the mean is

$$\bar{x}_1 = \frac{1}{N} \sum_{n=1}^{N} x_1^{(n)}$$

, with properties

$$E(\bar{x}_1) = E(x_1)$$

$$V(\bar{x}_1) = \frac{1}{N} V(x_1)$$

# Random variables

## CONDITIONALITY

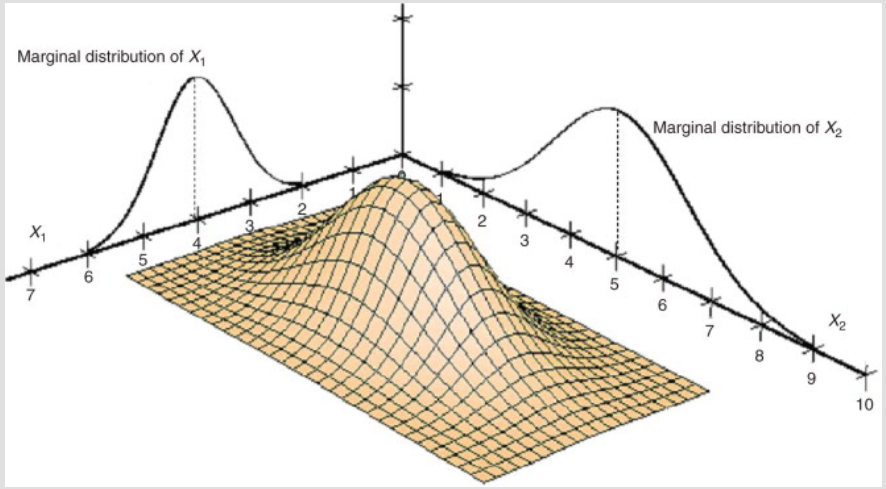
- Conditionality is a key concept in statistics; for a bivariate distribution, the chain rule is:

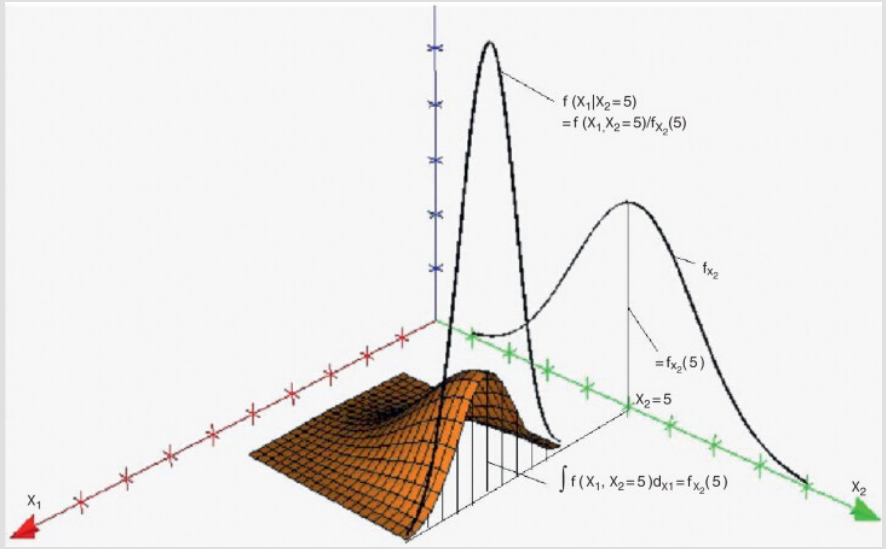
$$p(x_1|x_2) = \frac{p(x_1, x_2)}{p(x_2)}$$

$$= \frac{p(x_1, x_2)}{\int p(x_1, x_2) dx_2}$$

- The conditional expectation is used in many loss functions in machine learning;

$$E(x_1|x_2) = \int_{x_1} x_1 \; p(x_1|x_2) dx_1$$



# Random variables

## BAYES' THEOREM

- Bayes' theorem uses the chain rule (conditionality)

$$p(x_1, x_2) = p(x_2)p(x_1|x_2)$$

twice, turning the conditionality around.

- Say that we have observed $p(x_1|x_2)$ and have prior the prior density $p(x_2)$, then,

$$p(x_2|x_1) = \frac{p(x_2)p(x_1|x_2)}{p(x_1)}$$
$$\propto p(x_2)p(x_1|x_2)$$

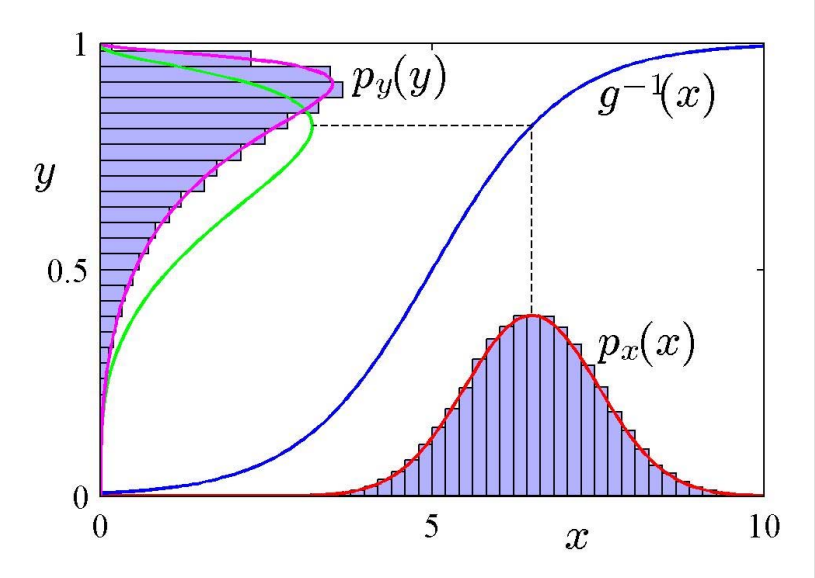
LUND UNIVERSITY

# Random variables

## TRANSFORMATION THEOREM

- The density of some variable $y$ may be expressed using the density of another variable, $x$ .
- This requires the mapping from y to x, $g : y \rightarrow x$, to be a monotonic function. Then,

$$p(y) = \left| \frac{d}{dy} \left( g^{-1}(y) \right) \right| \; p\left( g^{-1}(y) \right)$$




https://openai.com/blog/glow/

# Random variables

## LIKELIHOOD FUNCTION

- Say that we have a probabilistic model indexed by some variables $\mathbf{w}$, and a dataset $\mathbf{t}$
- The likelihood function for a dataset of observed samples is its joint density conditional on the data. It measures how likely it is to have observed our data if it truly was sampled from the model.

$$\mathcal{L}(\mathbf{t}|\mathbf{w}) = p(\mathbf{t}|\mathbf{w}) \overset{\text{i.i.d.}}{=} \prod_{n=1}^{N} p(t_n)|\mathbf{w})$$

- The maximum likelihood approach estimates the parameters $\mathbf{w}$ by minimizing the negative log-likelihood, i.e., for independent samples,

$$\underset{\mathbf{w}}{\text{minimize}} \; -\sum_{n=1}^{N} \ln p(t_n|\mathbf{w})$$

LUND UNIVERSITY

# Random variables

## GAUSSIAN DISTRIBUTION

- A multivariate Gaussian variable is completely defined by its expectation and covariance matrix, i.e.,

$$\mathbf{t} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$E(\mathbf{t}) = \boldsymbol{\mu} \in \mathbb{R}^N$$

$$V(\mathbf{t}) = \boldsymbol{\Sigma} \in \mathbb{R}^{N \times N}$$

$$\{\boldsymbol{\Sigma}\}_{i,j} = C(t_i, t_j)$$

- The multivariate Gaussian's density is

$$p(\mathbf{t}) = \frac{1}{(2\pi)^{N/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{t}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{t}-\boldsymbol{\mu})\right)$$

LUND UNIVERSITY

# Random variables

## GAUSSIAN DISTRIBUTION

- Any Gaussian variable may be rewritten as an affine transformation of an uncorrelated zero mean unit variance Gaussian variable,

$$\mathbf{t} = \boldsymbol{\Sigma}^{1/2}\mathbf{e} + \boldsymbol{\mu}$$

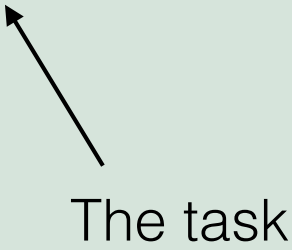
where the density for the (standardized) Gaussian is

$$p(\mathbf{e}) = \frac{1}{(2\pi)^{N/2}} \exp\left(-\frac{1}{2}\mathbf{e}^{\top}\mathbf{e}\right)$$

LUND UNIVERSITY

# Linear Regression

# Regression model (T)



The task

LUND UNIVERSITY

# Regression model

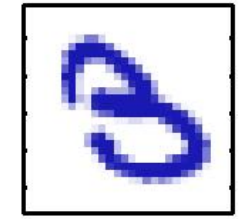
## NOTATIONAL CONVENTIONS

- Recall how notation is used in machine learning problems:

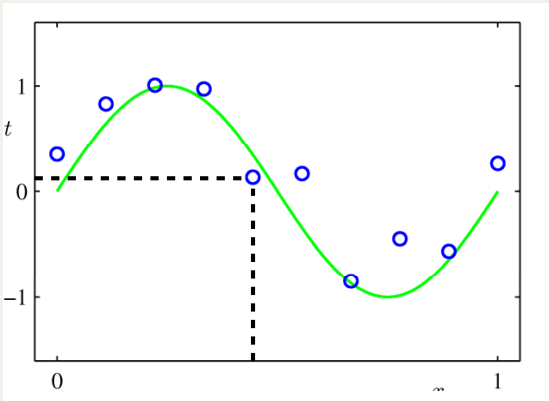
| | |
|---|---|
| $t_i, \mathbf{t}, \mathbf{T}$ | Targets |
| $x_i, \mathbf{x}, \mathbf{X}$ | Inputs |
| $y_i, \mathbf{y}, \mathbf{Y}$ | Model output |
| $w_i, \mathbf{w}, \mathbf{W}$ | trainable variables |
| $\lambda, \mu, \rho, \ldots$ | (capacity) hyperparameters |

= Data

no targets = unsupervised data

$\mathbf{X_i} =$ 

$t_i = 3$



# Linear Models

- It is mathematically easy to fit linear models to data and we can get insight by analyzing this relatively simple case. We already studied the polynomial curve fitting problem in the previous lecture
- There are many ways to make linear models more powerful while retaining their attractive mathematical properties
- By using non-linear, non-adaptive basis functions, we obtain generalized linear models. These offer non-linear mappings from inputs to outputs but are linear in their parameters. Typically, only the linear part of the model is learnt

# Linear Basis Function Models (2)

Generally

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

where $\phi_j(x)$ are known as *basis functions*.

- Typically, $\phi_0(x) = 1$ (dummy basis function) so that $w$ acts as a bias (allows for any fixed offset in the data – do not confuse with statistical bias)
- In the simplest case, we use linear basis functions: $\phi_d(x) = x_d$

# Regression model

## NOTATIONAL CONVENTIONS

$$y_i = \boldsymbol{\phi}(\mathbf{x}_i)^\top \mathbf{w}$$

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

$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\phi}(\mathbf{x}_1)^\top \\ \vdots \\ \boldsymbol{\phi}(\mathbf{x}_N)^\top \end{bmatrix}$$

Bishop (PR and ML)

$$\mathbf{y} = \mathbf{X}\mathbf{w}$$

Goodfellow (Deep Learning)

$$\boldsymbol{\Phi} = \mathbf{X} \in \mathbb{R}^{(N \times M)}$$

Number of features in hypothesis space

Number of data examples

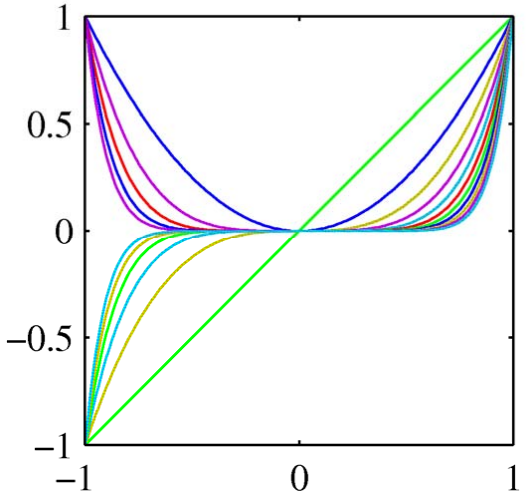
LUND UNIVERSITY

# Linear Basis Function Models (3)

Polynomial basis functions:

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

These are global; a small change in $x$ affects responses of all basis functions.



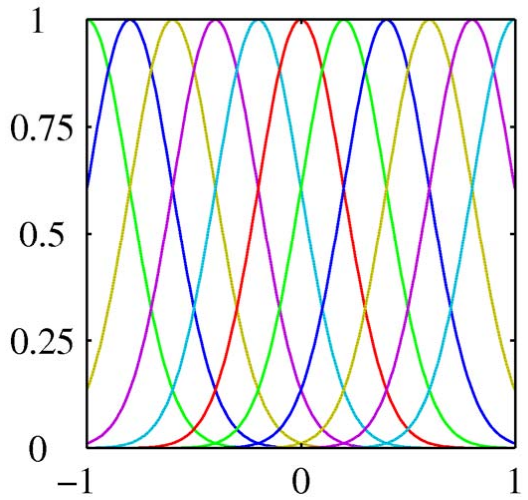

The issue can be resolved by dividing input space into regions and fitting a different polynomial in each region. This leads to *spline functions*.

# Linear Basis Function Models (4)

Gaussian basis functions:

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

These are local; a small change in $x$ only affect nearby basis functions. $\mu_j$ and $s$ control location and scale (width).



# Linear Basis Function Models (5)

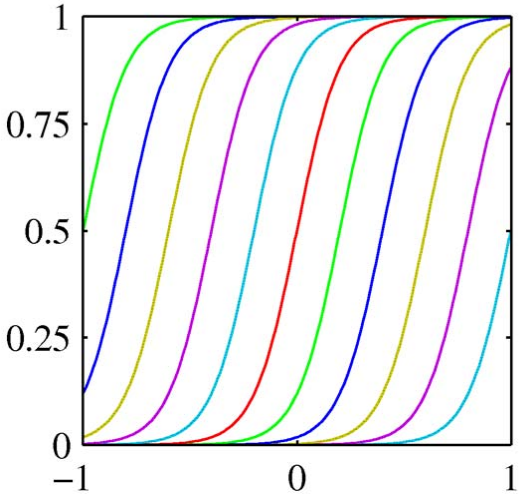
Sigmoidal basis functions:

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

where

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

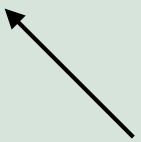
Also these are local; a small change in $x$ only affect nearby basis functions. $\mu_j$ and $s$ control location and scale (slope).



# Other Basis Function Models (6)

- In a Fourier representation, each basis function represents a given frequency and has infinite spatial extent
- Wavelets are localized in both space and frequency, and by definition are linearly orthogonal

# Loss function (P)



The performance metric

LUND UNIVERSITY

# Loss function

- In linear regression, the common practice is to use the mean squared error as loss function, i.e.,

$$MSE(\mathbf{w}) = N^{-1} \sum_{n=1}^{N} (t_i - \boldsymbol{\phi}(\mathbf{x}_i)^\top \mathbf{w})^2 = N^{-1} ||\mathbf{t} - \boldsymbol{\Phi}\mathbf{w}||_2^2$$

- Why is it so commonly used?

→ MSE is maximum likelihood when the targets are deterministic samples from the hypothesis space plus an additive Gaussian noise.

LUND UNIVERSITY

# Loss function

## Maximum Likelihood and Least Squares (1)

Assume observations from a deterministic function with added Gaussian noise:

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

which is the same as saying,

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1}) = (\beta/2\pi)^{\frac{1}{2}} \exp\{-\frac{\beta}{2}\big(t - y(x, w)\big)^2\}$$

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

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

# Loss function

## 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

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

is the sum-of-squares error

# Loss function

## Maximize Probabilities or their Log?

- We have a statistical model of outputs
- Assume the output errors on different training cases, $i$, are independent
- We will consider the product of the probabilities of the outputs on the training cases

$$p(\mathrm{t} \mid \mathrm{x}, w) = \prod_i p(t_i \mid x_i, w)$$

- Because the log function is monotonic, it does not change where the maxima are. Therefore, we can maximize the sum of log probabilities, or minimize negative log probabilities

$$L(\mathrm{x}, \mathrm{t}, w) = -\log p(\mathrm{t}, \mathrm{x} \mid w) = -\sum_i \log p(t_i \mid x_i, w)$$

# Loss function

## 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 $\mathbf{w}$, we get

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

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

where

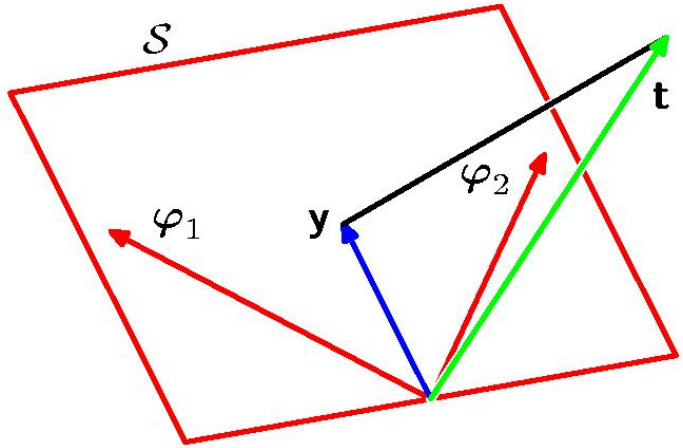
$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N)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

***design matrix (NxM)***

# Loss function

## Geometry of Least Squares

- The space has one axis for each training case. The vector of target values, $\mathbf{t} = (t_1, t_2 \ldots t_N)^T$ is a point in space
- Each vector of the values of one component of the input $\varphi_j$ is also a point in this space.
- The input component vectors span a subspace, $S$. A weighted sum of the input component vectors must lie in $S$.
- The optimal solution is the **orthogonal projection** of the vector of target values onto $S$.




$$\mathbf{y} = \mathbf{\Phi}\mathbf{w}_{\mathrm{ML}} = [\varphi_1, \ldots, \varphi_M]\,\mathbf{w}_{\mathrm{ML}}.$$

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

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

$\varphi_1(\mathbf{x})$ — second column; $\phi(\mathbf{x}_2)$ — second row

# Regularization (P)

# Learning: General Objective Functions

- The general structure of our learning objective function is

$$f(x, t, w) = L(x, t, w) + R(w)$$

$L$ is the loss function, and $R$ is a regularizer (penalty, or prior over functions), which discourages overly complex models

- **Intuition**
  - It is good to fit the data well and achieve low training loss
  - But it is also good to bias the machine towards simpler models, in order to avoid overfitting

- Setup allows to decouple optimization from choice of training loss

# Regularized Least Squares (1)

Consider the error function:

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

Data term + Regularization term

With the sum-of-squares error function and a quadratic regularizer, we get

$$\underset{\mathbf{w}}{\text{minimize}} \ \frac{1}{2}||\mathbf{t} - \mathbf{\Phi w}||_2^2 + \frac{\lambda}{2}||\mathbf{w}||_2^2$$

which is minimized by

$$\mathbf{w} = \left(\lambda \mathbf{I} + \mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1} \mathbf{\Phi}^{\mathrm{T}}\mathbf{t}.$$

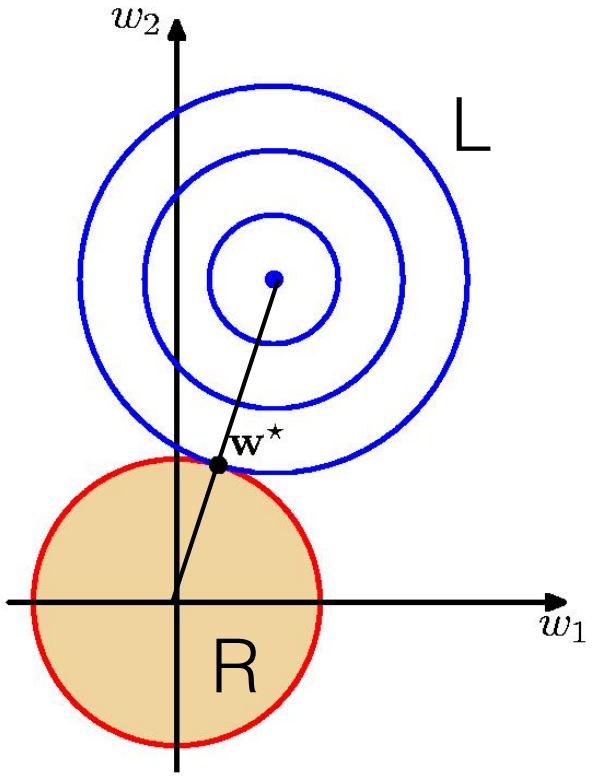
# Regularization

## RIDGE REGRESSION

$$\underset{\mathbf{w}}{\text{minimize}} \; ||\mathbf{t} - \mathbf{\Phi w}||_2^2$$

$$\text{subject to} \quad ||\mathbf{w}||_2^2 \leq \mu$$

# Effect of a Quadratic Regularizer




- The overall cost function is the sum of two quadratic functions
- The sum is also a quadratic
- The combined minimum lies on the line between the minimum of the squared error and the origin
- The regularizer shrinks the weights, but does not make them exactly zero

LUND UNIVERSITY

# Regularization

## RIDGE REGRESSION

- May also be analyzed using its principal components:
- The SVD of the N x M regression matrix is:

$$\mathbf{\Phi} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$$

- The reconstruction of the MSE estimate may be expressed as:

$$\begin{aligned}\mathbf{y} &= \mathbf{\Phi}\mathbf{w} = \mathbf{\Phi}(\mathbf{\Phi}^\top\mathbf{\Phi})^{-1}\mathbf{\Phi}^\top\mathbf{t} \\ &= \mathbf{U}\mathbf{U}^\top\mathbf{t}\end{aligned}$$

- For Ridge regression, the corresponding reconstruction becomes:

$$\begin{aligned}\mathbf{y} &= \mathbf{\Phi}\mathbf{w} = \mathbf{\Phi}(\mathbf{\Phi}^\top\mathbf{\Phi} + \lambda\mathbf{I})^{-1}\mathbf{\Phi}^\top\mathbf{t} \\ &= \mathbf{U}\mathbf{D}(\mathbf{D}^{\mathbf{2}} + \lambda\mathbf{I})^{-\mathbf{1}}\mathbf{D}\mathbf{U}^\top\mathbf{t} \\ &= \sum_{i=1}^{N}\mathbf{u}_i\frac{d_i^2}{d_i^2+\lambda}\mathbf{u}_i^\top\mathbf{t}\end{aligned}$$

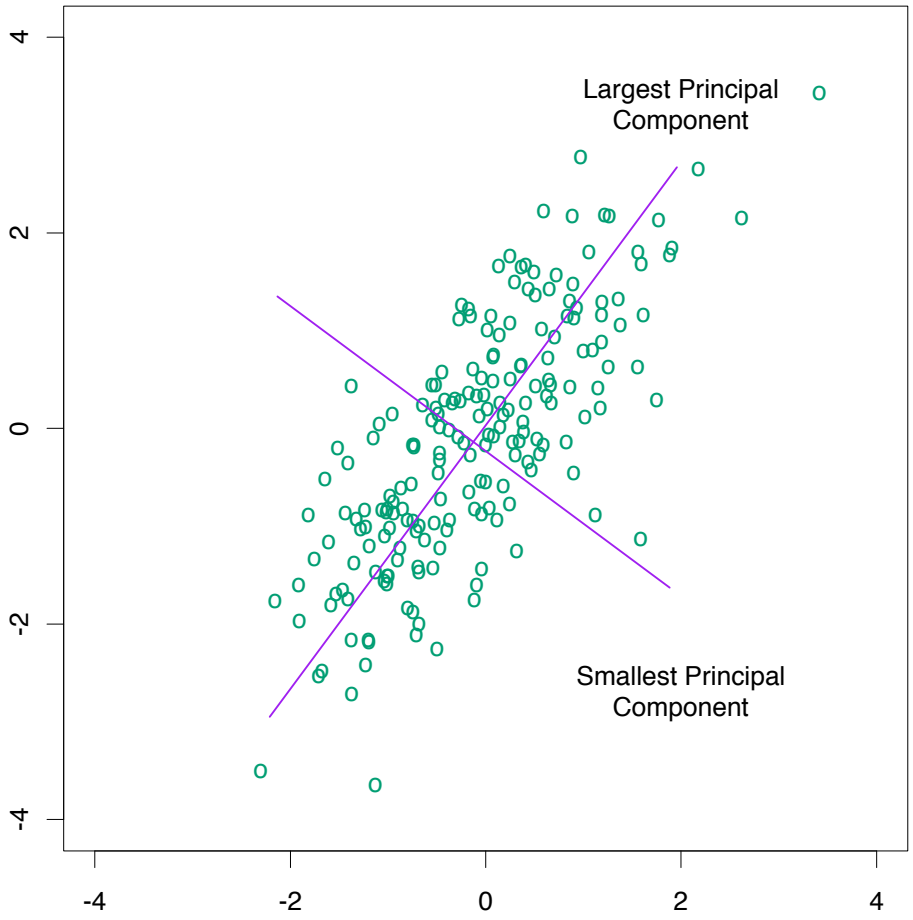
LUND UNIVERSITY

# Regularization

## RIDGE REGRESSION

- Ridge regression penalizes the smaller principal components more than the large ones.

$$\mathbf{y} = \sum_{i=1}^{N} \mathbf{u}_i \frac{d_i^2}{d_i^2 + \lambda} \mathbf{u}_i^\top \mathbf{t}$$



# Regularization

## LASSO

**Lasso**: penalizing the absolute values of weights

$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|$$

- Finding the minimum requires quadratic programming but it is still unique because the cost function is convex
  - Sum of two convex functions: a bowl plus an inverted pyramid
- As $\lambda$ is increased, many of the weights go exactly to zero
  - This is good for interpretation: identify factors causing a given disease, given input vector $\mathbf{x}$ containing many measurement components (temperature, blood pressure, sugar level, etc.)
  - It is also good in preventing overfitting

# Regularization

## LASSO

- The LASSO optimization problem may compactly be formulated as:

$$\underset{\mathbf{w}}{\text{minimize}} \ \frac{1}{2}||\mathbf{t} - \mathbf{X}\mathbf{w}||_2^2 + \lambda||\mathbf{w}||_1$$

- It has no closed-form solution like ridge regression.
- Optimizing one coordinate at a time, i.e.,

$$\underset{w_i}{\text{minimize}} \ \frac{1}{2}||\mathbf{r}_i - \mathbf{x}_i w_i||_2^2 + \lambda|w_i|, \quad \mathbf{r}_i = \mathbf{t} - \sum_{\ell \neq i} \mathbf{x}_\ell w_\ell$$

- A closed form expression is obtained as

$$\hat{w}_i = \frac{\text{sign}(\mathbf{x}_i^\top \mathbf{r}_i)}{\mathbf{x}_i^\top \mathbf{x}_i} \max(0, |\mathbf{x}_i^\top \mathbf{r}_i| - \lambda)$$

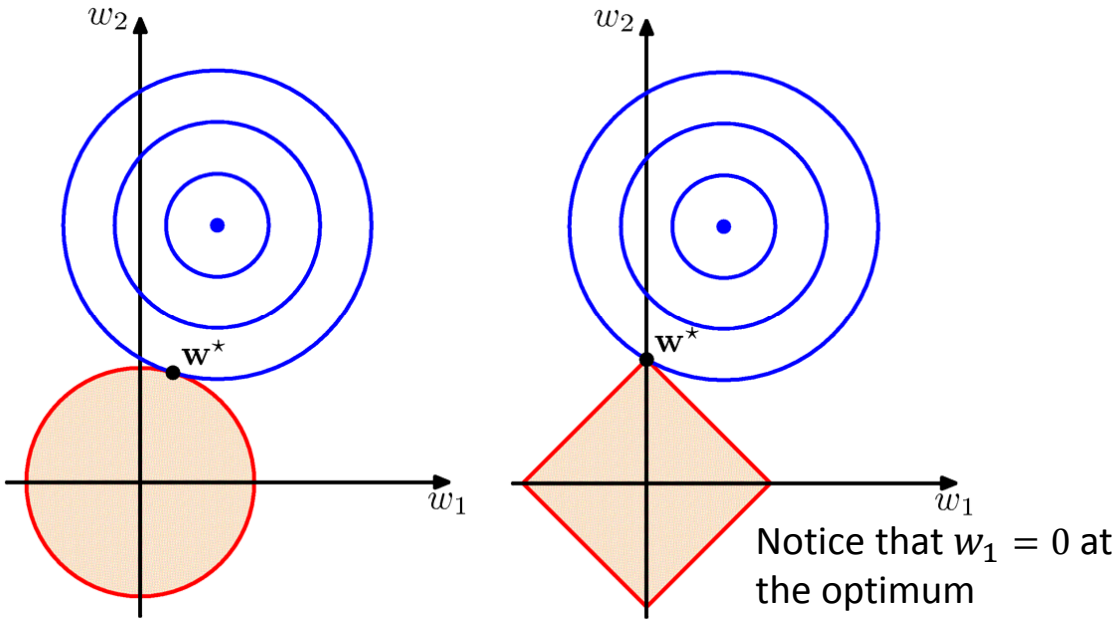
LUND UNIVERSITY

# Regularization

## LASSO

## Penalty over square weights vs. Lasso

Lasso tends to generate sparser solutions than a quadratic regularizer




Notice that $w_1 = 0$ at the optimum

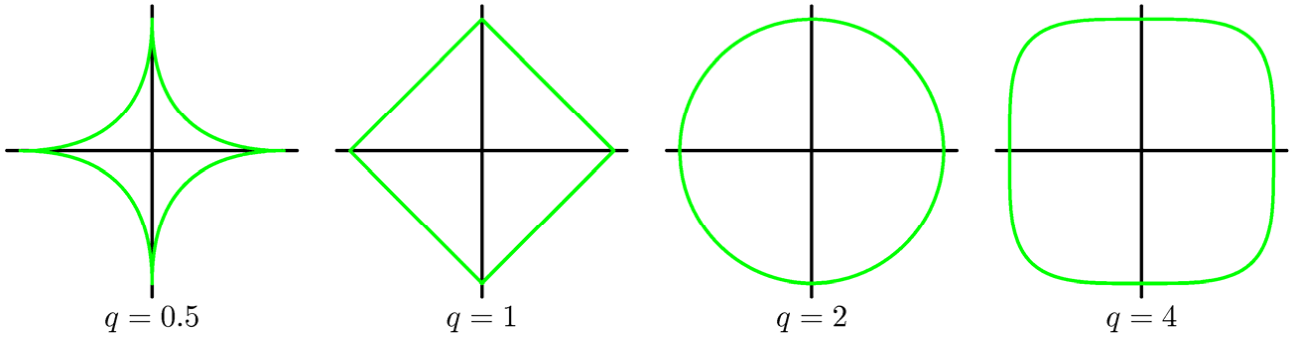
$$\underset{\mathbf{w}}{\text{minimize}} \; ||\mathbf{t} - \mathbf{X}\mathbf{w}||_2^2$$

$$\text{subject to} \quad ||\mathbf{w}||_1 \leq \mu$$

# Regularized Least Squares (2)

With a more general regularizer, we have

$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q$$




Lasso Quadratic

# Bias-variance decomposition and trade-off (P)

## Task (T)
### CAPACITY

”The central challenge in machine learning is that our algorithm must perform well on new, previously unseen inputs - not just those those on which our model was trained. The ability […] is called **generalization**”

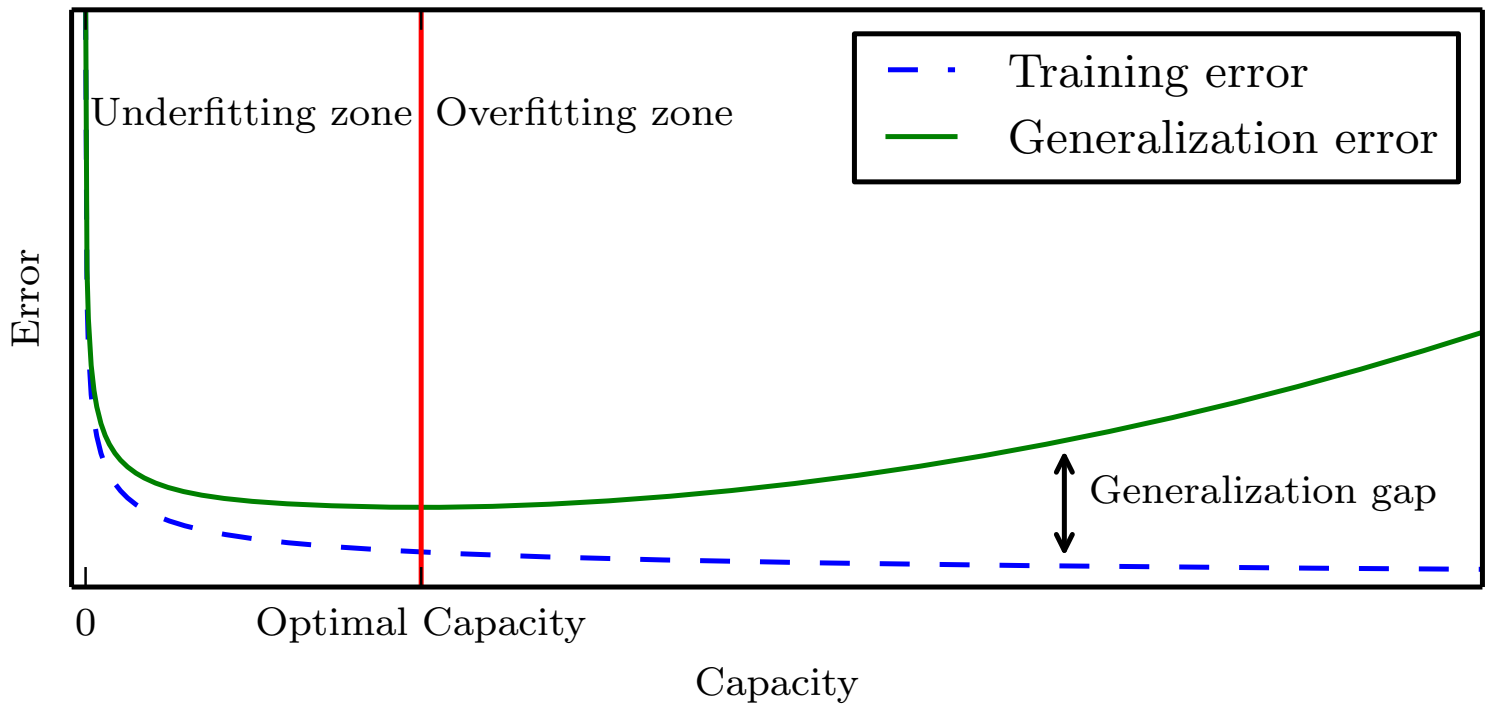



Figure 5.3

# Bias-variance decomposition

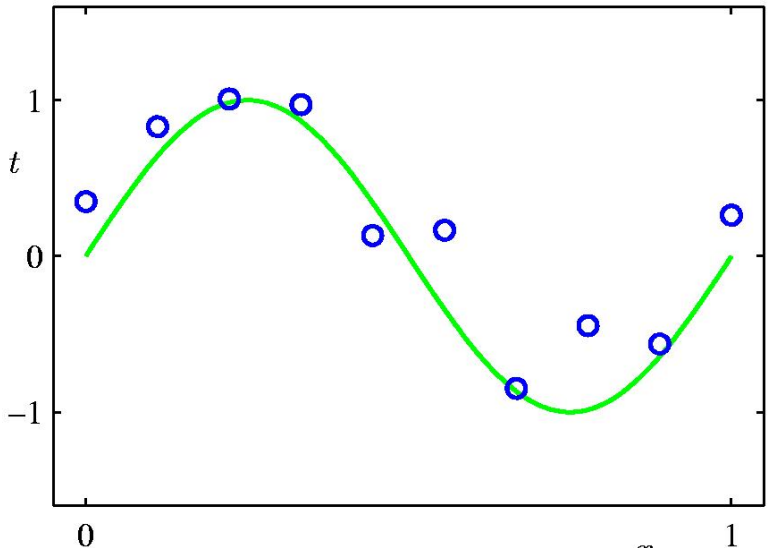
- The mean square error metric in machine learning problems may be understood by looking at its expectation.
- In the regression setting, we have:
  - Target value: $t = f(x) + e$
  - The estimated output: $y = \hat{f}(x) = \phi(x)^\top \mathbf{w}$
- And the expected MSE becomes:

$$
\begin{aligned}
E\left[(t-y)^2\right] &= E\left[\left(f(x) + e - \hat{f}(x)\right)^2\right] \\
&= E\left[\left(f(x) + e - \hat{f}(x) + E[\hat{f}(x)] - E[\hat{f}(x)]\right)^2\right] = \ldots \\
&= \left(f(x) - E[\hat{f}(x)]\right)^2 + E\left[\hat{f}(x) - E[\hat{f}(x)]\right]^2 + E\left[e^2\right] \\
&= \text{bias}(\hat{f}(x))^2 + \text{variance}(\hat{f}(x)) + \text{variance}(e)
\end{aligned}
$$

LUND UNIVERSITY

# Bias-variance decomposition

## Polynomial Curve Fitting




Synthetic data generated from:

$t(x) = sin2\pi x + \epsilon$,
$\epsilon \sim N\ (0, 0.3)$

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

Model is **linear** in the parameters *w* and **non-linear** in the input $x$

# Spline Fitting: Regularization

Penalize large coefficient values

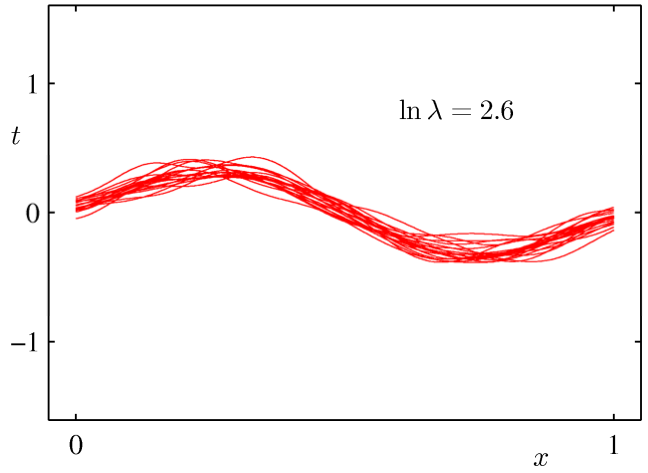
$$\underset{\mathbf{w}}{\text{minimize}} \ \frac{1}{2}||\mathbf{t} - \mathbf{\Phi w}||_2^2 + \frac{\lambda}{2}||\mathbf{w}||_2^2$$

Shrinkage approaches
For quadratic regularizers: ridge regression, weight decay (neural networks)

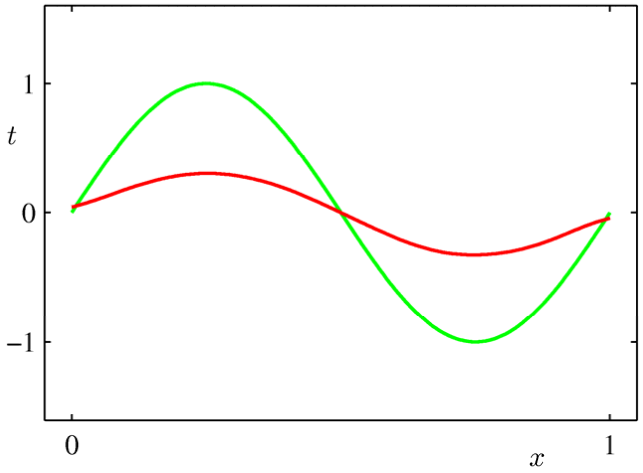
LUND UNIVERSITY

# The Bias-Variance Decomposition (5)

Example: 100 data sets, each with $N = 25$ datapoints, from sinusoidal, varying the degree of regularization, $\lambda$




Low variance

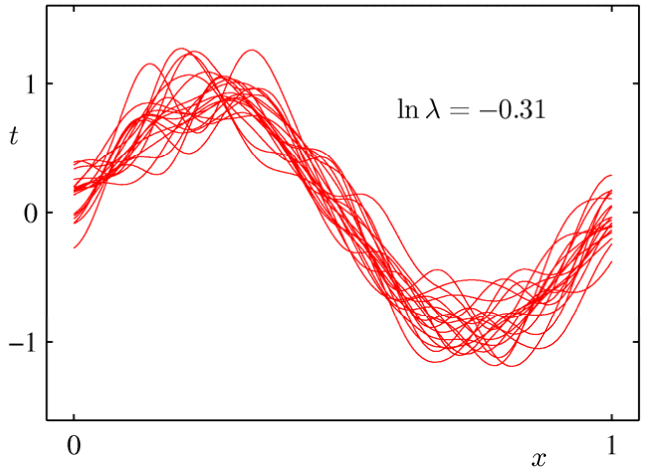



High bias

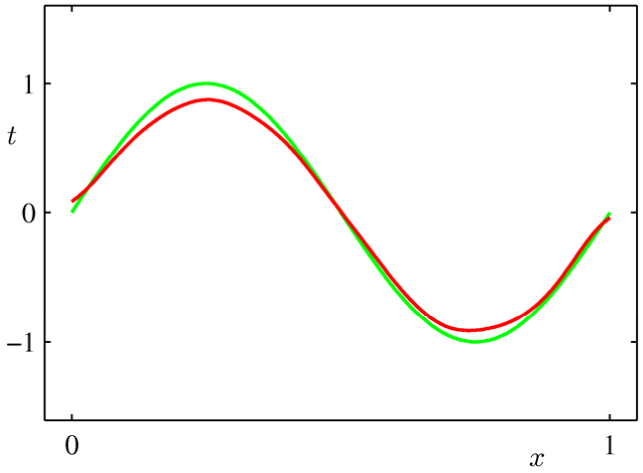
$$\underset{\mathbf{w}}{\text{minimize}} \ \frac{1}{2}||\mathbf{t} - \mathbf{\Phi w}||_2^2 + \frac{\lambda}{2}||\mathbf{w}||_2^2$$

# The Bias-Variance Decomposition (6)

Example: 100 data sets, each with $N = 25$ datapoints, from sinusoidal, varying the degree of regularization, $\lambda$




Moderate variance

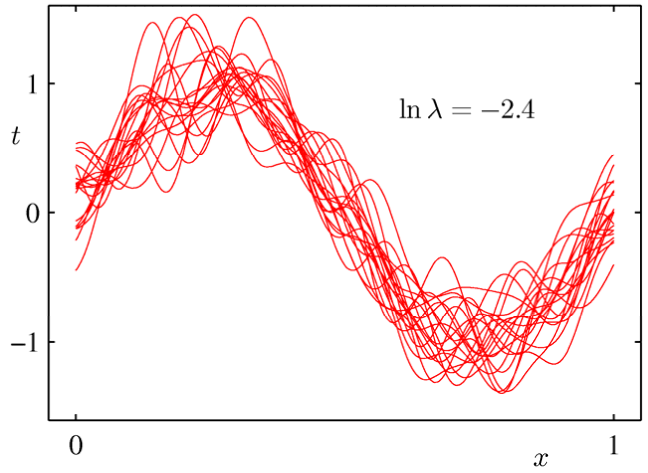



Moderate bias

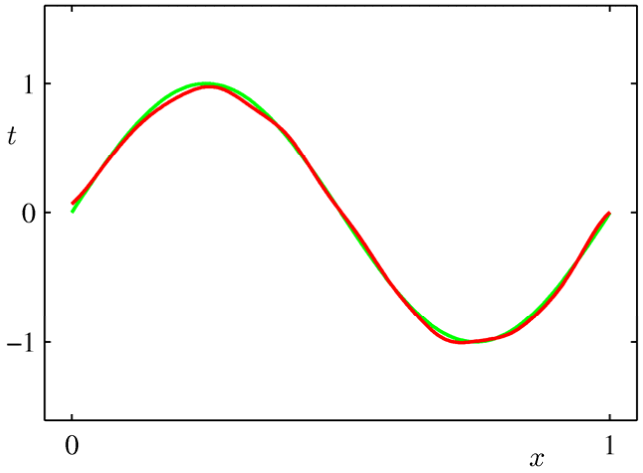
$$\underset{\mathbf{w}}{\text{minimize}} \ \frac{1}{2}||\mathbf{t} - \mathbf{\Phi w}||_2^2 + \frac{\lambda}{2}||\mathbf{w}||_2^2$$

# The Bias-Variance Decomposition (7)

Example: 100 data sets, each with $N = 25$ datapoints, from sinusoidal, varying the degree of regularization, $\lambda$



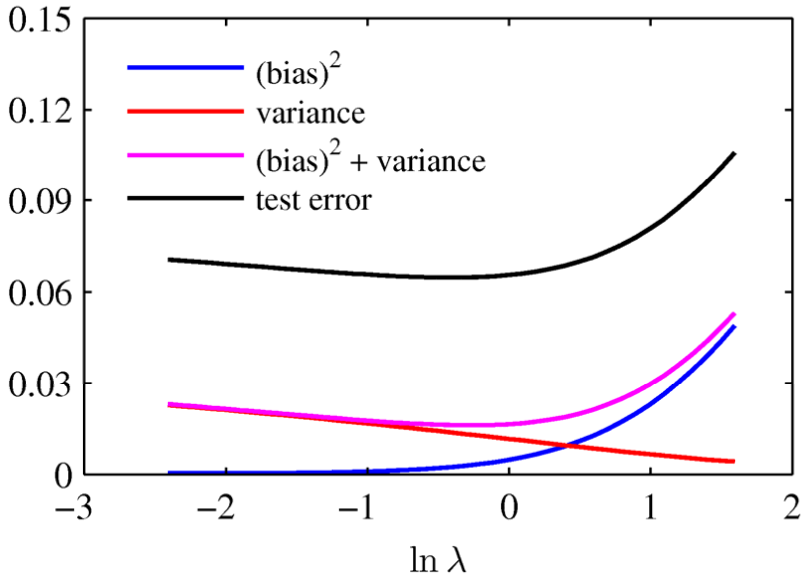

High variance




Low bias

$$\underset{\mathbf{w}}{\text{minimize}} \ \frac{1}{2}||\mathbf{t} - \mathbf{\Phi w}||_2^2 + \frac{\lambda}{2}||\mathbf{w}||_2^2$$

# The Bias-Variance Trade-off




- The test error may be estimated using, e.g., cross-validation
- Visualization is typically better with a log-scale on the x-axis.

From these plots, we note that an over-regularized model (large $\lambda$) will have a high bias, while an under-regularized model (small $\lambda$) will have a high variance. Minimum around $\ln \lambda = -0.31$

---

$$E\left[(t-y)^2\right] = \text{bias}(\hat{f}(x)) + \text{variance}(\hat{f}(x)) + \text{variance}(e)$$

# Bayesian regression (T) and (P)

LUND UNIVERSITY

# Frequentist: General Objective Functions

- The general structure of our learning objective function is

$$f(x, t, w) = L(x, t, w) + R(w)$$

$L$ is the loss function, and $R$ is a regularizer (penalty, or prior over functions), which discourages overly complex models

- If we assume that all parameter configurations are equally likely (otherwise said, the prior over $w$ is uniform, in the selected parameterization), we obtain **Maximum Likelihood (ML).** This selects model parameters that assign the highest probability to observed data
- If we assume a prior over models or parameters, we obtain a regularized ML estimate, the **Maximum a-posteriori estimate (MAP)**

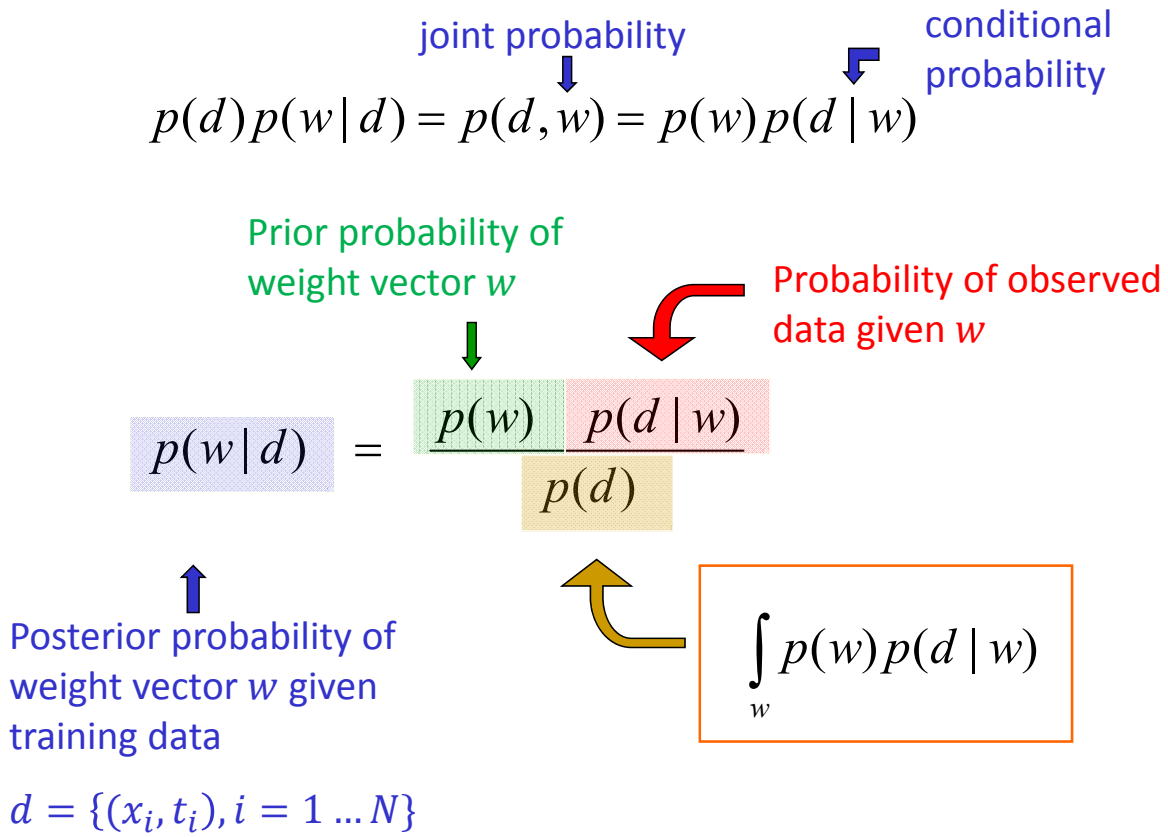
# Learning in a Bayesian Framework

The Bayesian framework assumes that we have a prior distribution over all variables of interest, and our goals is to compute probability distributions, not point estimates

- The prior may be vague
- We combine our prior distribution with the data likelihood term to construct the posterior distribution
- The likelihood accounts for how probable the observed data is given the model parameters
  - It favors parameter that make the data likely
  - It counteracts the prior
  - With enough data, the likelihood dominates

# Bayesian regression

## *Learning Setup:* Bayes' Theorem



# Bayesian regression

**Prior:**
$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}) = \left(\frac{\alpha}{2\pi}\right)^{(M+1)/2} \exp\left\{-\frac{\alpha}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}\right\}$$

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

**Posterior:**
$$p(\mathbf{w}|\mathbf{x}, \mathbf{t}, \alpha, \beta) \propto p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta)p(\mathbf{w}|\alpha)$$

$$\beta\widetilde{E}(\mathbf{w}) = \frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\alpha}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$$

Determine $\mathbf{w}_{\mathrm{MAP}}$ by minimizing regularized sum-of-squares error, $\widetilde{E}(\mathbf{w})$ with regularizer corresponding to $\lambda = \alpha/\beta$

LUND UNIVERSITY

## Self study:

- **Goodfellow et al 2016 (Deep Learning):**
  - Chapter 3: up to 3.13
  - Chapter 5: 5.5-5.6
- **Bishop 2006 (Pattern Recognition and Machine Learning)**
  - Chapter 1: 1.2
  - Chapter 2: 2.1, 2.3
  - Chapter 3: 3.1-3.4
- **Hastie et al. 2019 (Elements of Statistical Learning)**
  - Chapter 2: 2.6
  - Chapter 3, up to 3.2.1 and 3.4.

LUND UNIVERSITY

# L-2: Probability and Regression

Ted Kronvall, Cristian Sminchisescu