Advanced Machine Learning
Summer 2019

Part 4 – Linear Regression III
10.04.2019

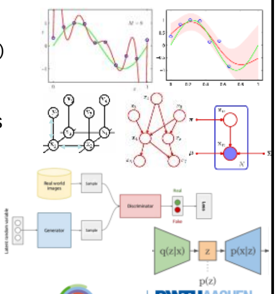
Prof. Dr. Bastian Leibe

RWTH Aachen University, Computer Vision Group
http://www.vision.rwth-aachen.de

Course Outline

$f : \mathcal{X} \rightarrow \mathbb{R}$

- Regression Techniques
  - Linear Regression
  - Regularization (Ridge, Lasso)
  - Kernels (Kernel Ridge Regression)
- Deep Reinforcement Learning
- Probabilistic Graphical Models
  - Bayesian Networks
  - Markov Random Fields
  - Inference (exact & approximate)
- Deep Generative Models
  - Generative Adversarial Networks
  - Variational Autoencoders



Topics of This Lecture

- Recap: Linear Regression
- Bias-Variance Trade-Off
- Kernels
  - Dual representations
  - Kernel Ridge Regression
  - Properties of kernels
- Other Kernel Methods
  - Kernel PCA
  - Kernel k-Means Clustering

Recap: Other Loss Functions for Regression

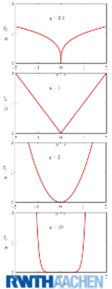
- The squared loss is not the only possible choice
  - Poor choice when conditional distribution $p(t|\mathbf{x})$ is multimodal.
- Simple generalization: Minkowski loss

$$L(t, y(\mathbf{x})) = |y(\mathbf{x}) - t|^q$$

  - Expectation

$$\mathbb{E}[L_q] = \iint |y(\mathbf{x}) - t|^q p(\mathbf{x}, t)\mathrm{d}\mathbf{x}\mathrm{d}t$$

- Minimum of $\mathbb{E}[L_q]$ is given by
  - Conditional mean for $q = 2$,
  - Conditional median for $q = 1$,
  - Conditional mode for $q = 0$.

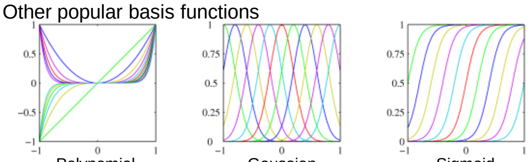


Recap: Linear Basis Function Models

- Generally, we consider models of the following form

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

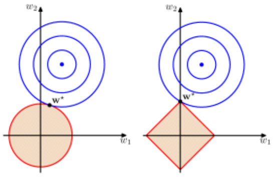
  - where $\phi_j(\mathbf{x})$ are known as *basis functions*.
  - In the simplest case, we use linear basis functions: $\phi_d(\mathbf{x}) = x_d$.
- Other popular basis functions



Polynomial Gaussian Sigmoid

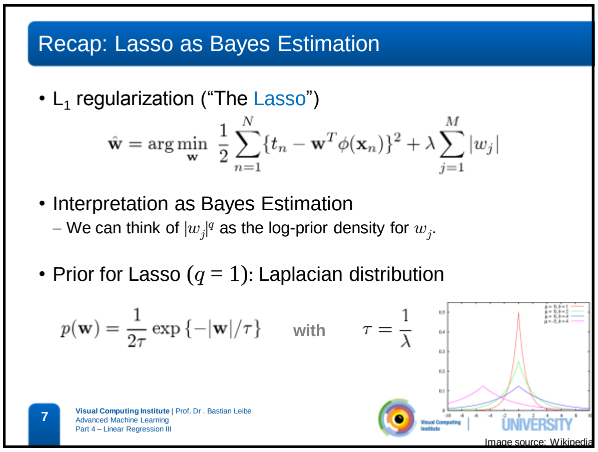
Recap: Regularized Least-Squares

- Consider more general regularization functions
  - "L$_q$ norms": $\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q$



- Effect: Sparsity for $q \leq 1$.
  - Minimization tends to set many coefficients to zero

Image source: C.M. Bishop, 2006



Recap: Lasso as Bayes Estimation

- L_1 regularization (“The Lasso”)

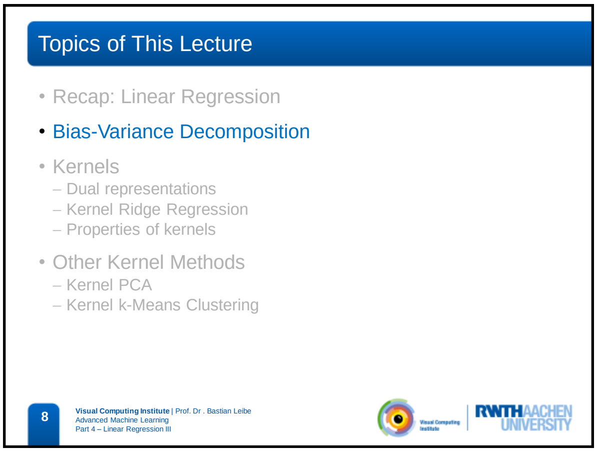
$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \ \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^T\phi(\mathbf{x}_n)\}^2 + \lambda\sum_{j=1}^{M}|w_j|$$

- Interpretation as Bayes Estimation
  - We can think of $|w_j|^q$ as the log-prior density for w_j.
- Prior for Lasso ($q = 1$): Laplacian distribution

$$p(\mathbf{w}) = \frac{1}{2\tau}\exp\{-|\mathbf{w}|/\tau\} \quad \text{with} \quad \tau = \frac{1}{\lambda}$$

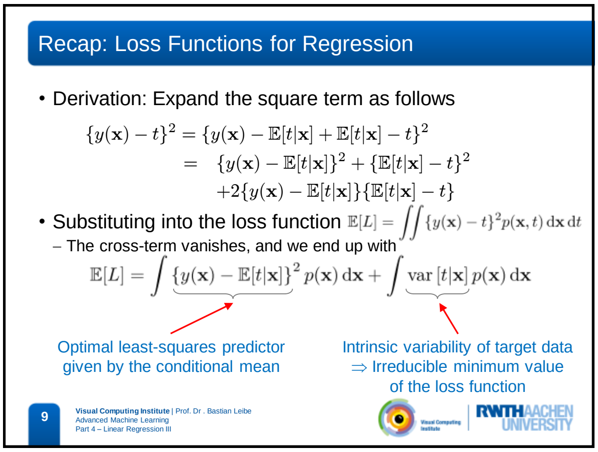
μ= 0, b=1
μ= 0, b=2
μ= 0, b=4
μ=-5, b=4
0.5
0.4
0.3
0.2
0.1
0
-10
-8
-6
-4
-2
0
2
4
6
8
10

Image source: Wikipedia



Topics of This Lecture

- Recap: Linear Regression
- Bias-Variance Decomposition
- Kernels
  - Dual representations
  - Kernel Ridge Regression
  - Properties of kernels
- Other Kernel Methods
  - Kernel PCA
  - Kernel k-Means Clustering



Recap: Loss Functions for Regression

- Derivation: Expand the square term as follows

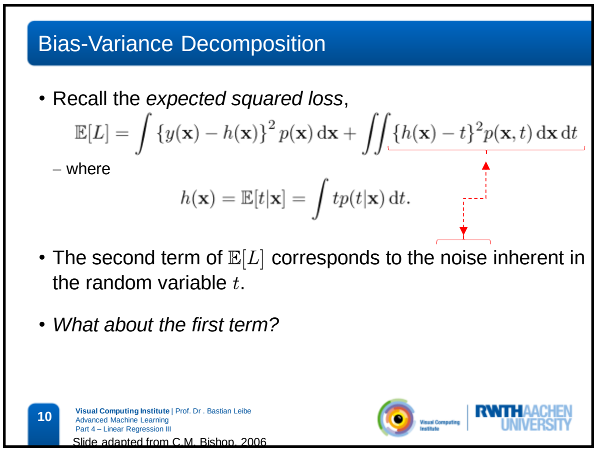
$$\begin{aligned}\{y(\mathbf{x}) - t\}^2 &= \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}] + \mathbb{E}[t|\mathbf{x}] - t\}^2 \\ &= \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}^2 + \{\mathbb{E}[t|\mathbf{x}] - t\}^2 \\ &\quad +2\{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}\{\mathbb{E}[t|\mathbf{x}] - t\}\end{aligned}$$

- Substituting into the loss function $\mathbb{E}[L] = \iint \{y(\mathbf{x}) - t\}^2 p(\mathbf{x}, t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t$
  - The cross-term vanishes, and we end up with

$$\mathbb{E}[L] = \int \underbrace{\{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}^2}\, p(\mathbf{x})\,\mathrm{d}\mathbf{x} + \int \underbrace{\mathrm{var}\,[t|\mathbf{x}]}\, p(\mathbf{x})\,\mathrm{d}\mathbf{x}$$

Optimal least-squares predictor given by the conditional mean

Intrinsic variability of target data ⇒ Irreducible minimum value of the loss function



Bias-Variance Decomposition

- Recall the *expected squared loss*,

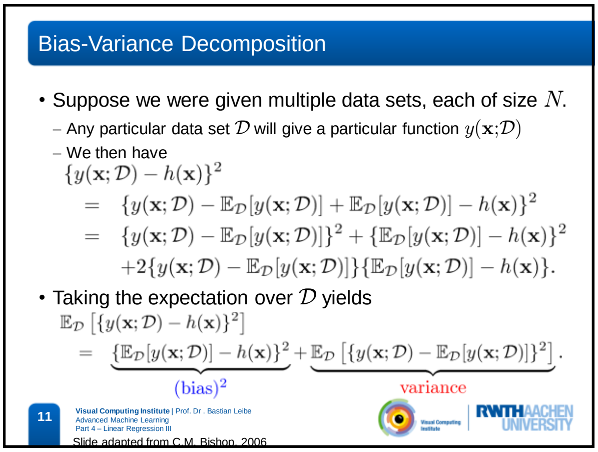
$$\mathbb{E}[L] = \int \{y(\mathbf{x}) - h(\mathbf{x})\}^2 p(\mathbf{x})\,\mathrm{d}\mathbf{x} + \iint \underbrace{\{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t}$$

  - where

$$h(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}] = \int t p(t|\mathbf{x})\,\mathrm{d}t.$$

- The second term of $\mathbb{E}[L]$ corresponds to the noise inherent in the random variable t.
- *What about the first term?*

Slide adapted from C.M. Bishop, 2006



Bias-Variance Decomposition

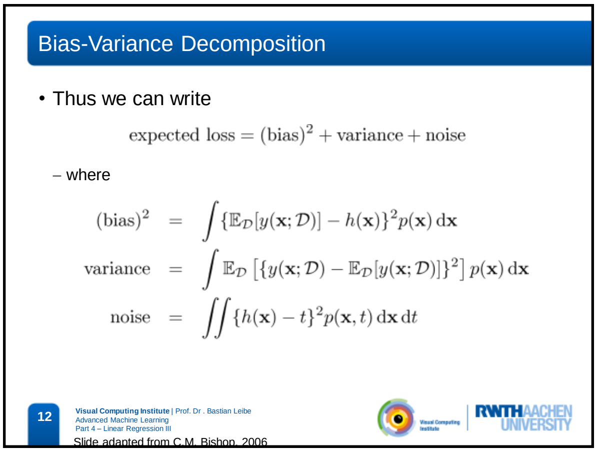
- Suppose we were given multiple data sets, each of size N.
  - Any particular data set $\mathcal{D}$ will give a particular function $y(\mathbf{x};\mathcal{D})$
  - We then have

$$\begin{aligned}&\{y(\mathbf{x};\mathcal{D}) - h(\mathbf{x})\}^2 \\ &= \{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] + \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2 \\ &= \{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2 + \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2 \\ &\quad +2\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}.\end{aligned}$$

- Taking the expectation over $\mathcal{D}$ yields

$$\begin{aligned}&\mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - h(\mathbf{x})\}^2\right] \\ &= \underbrace{\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2}_{(\mathrm{bias})^2} + \underbrace{\mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2\right]}_{\mathrm{variance}}.\end{aligned}$$

Slide adapted from C.M. Bishop, 2006



Bias-Variance Decomposition

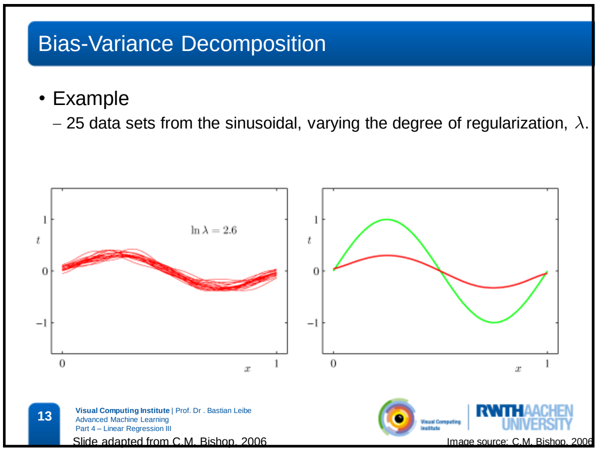
- Thus we can write

$$\text{expected loss} = (\text{bias})^2 + \text{variance} + \text{noise}$$

  - where

$$\begin{aligned}(\text{bias})^2 &= \int \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2 p(\mathbf{x})\,\mathrm{d}\mathbf{x} \\ \text{variance} &= \int \mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2\right] p(\mathbf{x})\,\mathrm{d}\mathbf{x} \\ \text{noise} &= \iint \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t\end{aligned}$$

Slide adapted from C.M. Bishop, 2006

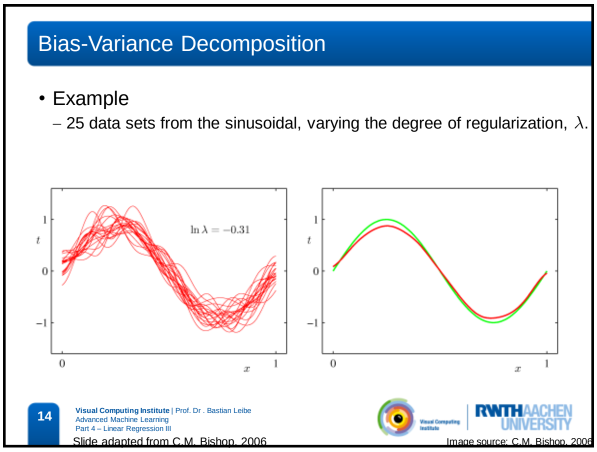


Bias-Variance Decomposition

- Example
  - 25 data sets from the sinusoidal, varying the degree of regularization, λ.

Slide adapted from C.M. Bishop, 2006

Image source: C.M. Bishop, 2006

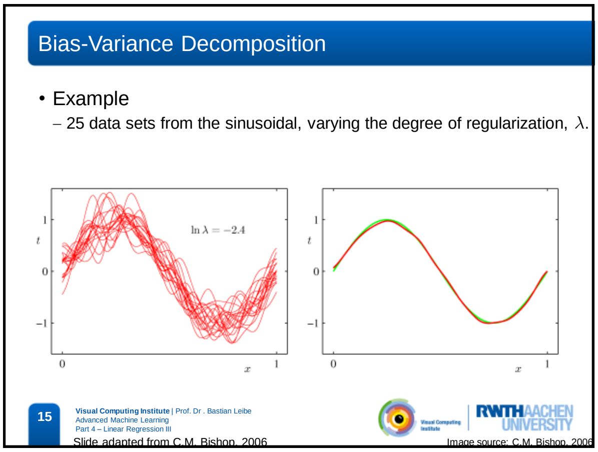


Bias-Variance Decomposition

- Example
  - 25 data sets from the sinusoidal, varying the degree of regularization, λ.

Slide adapted from C.M. Bishop, 2006

Image source: C.M. Bishop, 2006

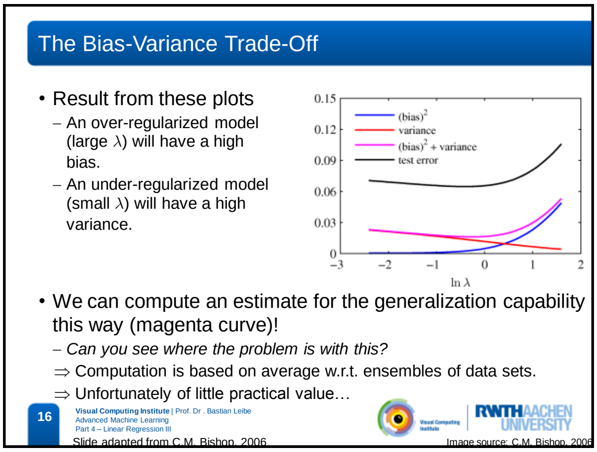


Bias-Variance Decomposition

- Example
  - 25 data sets from the sinusoidal, varying the degree of regularization, λ.

Slide adapted from C.M. Bishop, 2006

Image source: C.M. Bishop, 2006



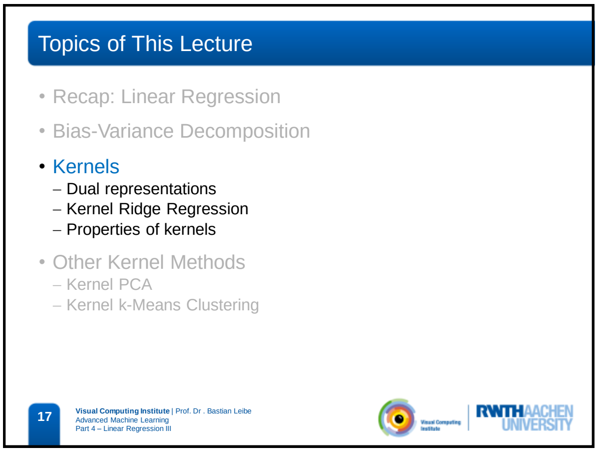
The Bias-Variance Trade-Off

- Result from these plots
  - An over-regularized model (large λ) will have a high bias.
  - An under-regularized model (small λ) will have a high variance.

- We can compute an estimate for the generalization capability this way (magenta curve)!
  - *Can you see where the problem is with this?*
  - ⇒ Computation is based on average w.r.t. ensembles of data sets.
  - ⇒ Unfortunately of little practical value…

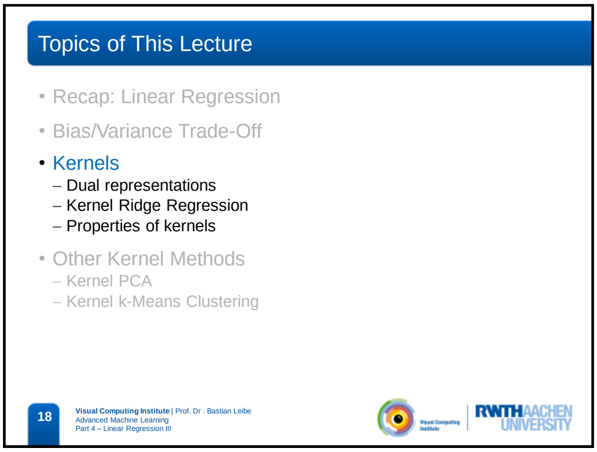
Slide adapted from C.M. Bishop, 2006

Image source: C.M. Bishop, 2006



Topics of This Lecture

- Recap: Linear Regression
- Bias-Variance Decomposition
- Kernels
  - Dual representations
  - Kernel Ridge Regression
  - Properties of kernels
- Other Kernel Methods
  - Kernel PCA
  - Kernel k-Means Clustering



Topics of This Lecture

- Recap: Linear Regression
- Bias/Variance Trade-Off
- Kernels
  - Dual representations
  - Kernel Ridge Regression
  - Properties of kernels
- Other Kernel Methods
  - Kernel PCA
  - Kernel k-Means Clustering

Introduction to Kernel Methods

- Dual representations
  - Many linear models for regression and classification can be reformulated in terms of a dual representation, where predictions are based on linear combinations of a kernel function evaluated at training data points.
  - For models that are based on a fixed nonlinear feature space mapping $\phi(\mathbf{x})$, the kernel function is given by

$$k(\mathbf{x},\mathbf{x}') = \phi(\mathbf{x})^T\phi(\mathbf{x}')$$

  - We will see that by substituting the inner product by the kernel, we can achieve interesting extensions of many well-known algorithms…

Dual Representations: Derivation

- Consider a regularized linear regression model

$$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{\mathbf{w}^T\phi(\mathbf{x}_n) - t_n\}^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

with the solution

$$\mathbf{w} \;=\; -\frac{1}{\lambda}\sum_{n=1}^{N}\{\mathbf{w}^T\phi(\mathbf{x}_n) - t_n\}\phi(\mathbf{x}_n)$$

  - We can write this as a linear combination of the $\phi(\mathbf{x}_n)$ with coefficients that are functions of $\mathbf{w}$:

$$\mathbf{w} \;=\; \sum_{n=1}^{N} a_n\phi(\mathbf{x}_n) = \Phi^T\mathbf{a}$$

with

$$a_n = -\frac{1}{\lambda}\{\mathbf{w}^T\phi(\mathbf{x}_n) - t_n\}$$

Dual Representations: Derivation

- Dual definition
  - Instead of working with $\mathbf{w}$, we can formulate the optimization for $\mathbf{a}$ by substituting $\mathbf{w} = \mathbf{\Phi}^T\mathbf{a}$ into $J(\mathbf{w})$:

$$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{\mathbf{w}^T\phi(\mathbf{x}_n) - t_n\}^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^T\Phi\Phi^T\Phi\Phi^T\mathbf{a} - \mathbf{a}^T\Phi\Phi^T\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^T\Phi\Phi^T\mathbf{a}$$

  - Define the kernel matrix $\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^T$ with elements

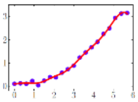
$$K_{nm} = \phi(\mathbf{x}_n)^T\phi(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m)$$

  - Now, the sum-of-squares error can be written as

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^T\mathbf{K}\mathbf{K}\mathbf{a} - \mathbf{a}^T\mathbf{K}\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^T\mathbf{K}\mathbf{a}$$

Kernel Ridge Regression

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^T\mathbf{K}\mathbf{K}\mathbf{a} - \mathbf{a}^T\mathbf{K}\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^T\mathbf{K}\mathbf{a}$$



  - Solving for $\mathbf{a}$, we obtain

$$\mathbf{a} \;=\; (\mathbf{K} + \lambda\mathbf{I}_N)^{-1}\mathbf{t}$$

- Prediction for a new input $\mathbf{x}$:
  - Writing $\mathbf{k}(\mathbf{x})$ for the vector with elements $k_n(\mathbf{x}) = k(\mathbf{x}_n, \mathbf{x})$

$$y(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) = \mathbf{a}^T\Phi\phi(\mathbf{x}) = \mathbf{k}(\mathbf{x})^T(\mathbf{K} + \lambda\mathbf{I}_N)^{-1}\mathbf{t}$$

⇒ *The dual formulation allows the solution to be entirely expressed in terms of the kernel function* $k(\mathbf{x},\mathbf{x}')$.

⇒ The resulting form is known as Kernel Ridge Regression and allows us to perform non-linear regression.

Image source: Christoph Lampert

Why use $k(\mathbf{x},\mathbf{x}')$ instead of $\phi(\mathbf{x})^T\phi(\mathbf{x}')$?

1. Memory usage
   - Storing $\phi(\mathbf{x}_1),\ldots, \phi(\mathbf{x}_N)$ requires $O(NM)$ memory.
   - Storing $k(\mathbf{x}_1, \mathbf{x}_1),\ldots, k(\mathbf{x}_N, \mathbf{x}_N)$ requires $O(N^2)$ memory.
2. Speed
   - We might find an expression for $k(\mathbf{x}_i, \mathbf{x}_j)$ that is faster to evaluate than first forming $\phi(\mathbf{x})$ and then computing $\phi(\mathbf{x})^T\phi(\mathbf{x}')$.
   - Example: comparing angles ($x \in [0, 2\pi]$):

$$\begin{aligned}\langle\phi(x_i), \phi(x_j)\rangle &= \langle[\cos(x_i), \sin(x_i)], [\cos(x_j), \sin(x_j)]\rangle \\ &= \cos(x_i)\cos(x_j) + \sin(x_i)\sin(x_j) \\ k(x_i, x_j) &:= \cos(x_i - x_j)\end{aligned}$$

Slide credit: Christoph Lampert

Why use $k(\mathbf{x},\mathbf{x}')$ instead of $\phi(\mathbf{x})^T\phi(\mathbf{x}')$?

3. Flexibility
   - There are kernel functions $k(\mathbf{x}_i, \mathbf{x}_j)$ for which we know that a feature transformation ϕ exists, but we don't know what ϕ is.
   - This allows us to work with far more general similarity functions.
   - We can define kernels on strings, trees, graphs, …
4. Dimensionality
   - Since we no longer need to explicitly compute $\phi(\mathbf{x})$, we can work with high-dimensional (even infinite-dim.) feature spaces.

- *In the following, we take a closer look at the background behind kernels and at how to use them…*

Slide credit: Christoph Lampert

Properties of Kernels

- Definition (Positive Definite Kernel Function)
  - Let $\mathcal{X}$ be a non-empty set. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called definite kernel function, iff
  - k is symmetric, i.e. $k(x, x') = k(x', x)$ for all $x, x' \in \mathcal{X}$, and
  - for any set of points $x_1, \dots, x_n \in \mathcal{X}$, the matrix

$$K_{ij} = (k(x_i, x_j))_{i,j}$$

is positive (semi-)definite, i.e. for all vectors $\mathbf{x} \in \mathbb{R}^n$:

$$\sum_{i,j=1}^{N} \mathbf{x}_i K_{ij} \mathbf{x}_j \geq 0$$

Slide credit: Christoph Lampert

Hilbert Spaces

- Definition (Hilbert Space)
  - A Hilbert Space $\mathcal{H}$ is a vector space H with an *inner product* $\langle . , . \rangle_{\mathcal{H}}$, e.g. a mapping

$$\langle . , . \rangle_{\mathcal{H}} : H \times H \to \mathbb{R}$$

which is

  - symmetric: $\langle v, v' \rangle_{\mathcal{H}} = \langle v', v \rangle_{\mathcal{H}}$ for all $v, v' \in H$,
  - positive definite: $\langle v, v \rangle_{\mathcal{H}} \geq 0$ for all $v \in H$, where $\langle v, v \rangle_{\mathcal{H}} = 0$ only for $v = \mathbf{0} \in H$.
  - bilinear: $\langle av, v' \rangle_{\mathcal{H}} = a\langle v, v' \rangle_{\mathcal{H}}$ for $v \in H, a \in \mathbb{R}$
    $\langle v + v', v'' \rangle_{\mathcal{H}} = \langle v, v'' \rangle_{\mathcal{H}} + \langle v', v'' \rangle_{\mathcal{H}}$
  - We can treat a Hilbert space like some $\mathbb{R}^n$, if we only use concepts like *vectors*, *angles*, *distances*.
  - Note: $dim\mathcal{H} = \infty$ is possible!

Slide credit: Christoph Lampert

Properties of Kernels

- Theorem
  - *Let k: $\mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive definite kernel function. Then there exists a Hilbert Space $\mathcal{H}$ and a mapping $\varphi : \mathcal{X} \to \mathcal{H}$ such that*

$$k(x, x') = \langle (\phi(x), \phi(x') \rangle_{\mathcal{H}}$$

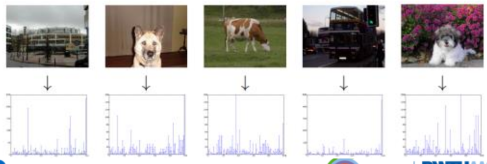
  - where $\langle . , . \rangle_{\mathcal{H}}$ *is the inner product in* $\mathcal{H}$.
- Translation
  - Take *any* set $\mathcal{X}$ and *any* function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.
  - If k is a positive definite kernel, then we can use k to learn a (soft) maximum-margin classifier for the elements in $\mathcal{X}$!
- Note
  - $\mathcal{X}$ can be any set, e.g. $\mathcal{X}$ = *"all videos on YouTube"* or $\mathcal{X}$ = *"all permutations of* {1, . . . , *k*}*"*, or $\mathcal{X}$ = *"the internet"*.

Slide credit: Christoph Lampert

Example: Bag of Visual Words Representation

- General framework in visual recognition
  - Create a codebook (vocabulary) of prototypical image features
  - Represent images as histograms over codebook activations
  - Compare two images by any histogram kernel, e.g. χ^2 kernel

$$k_{\chi^2}(h, h') = \exp\left(-\frac{1}{\gamma}\sum_j \frac{(h_j - h'_j)^2}{h_j + h'_j}\right)$$



Slide credit: Christoph Lampert

The "Kernel Trick"

Any algorithm that uses data only in the form of inner products can be *kernelized.*

- How to kernelize an algorithm
  - Write the algorithm only in terms of inner products.
  - Replace all inner products by kernel function evaluations.

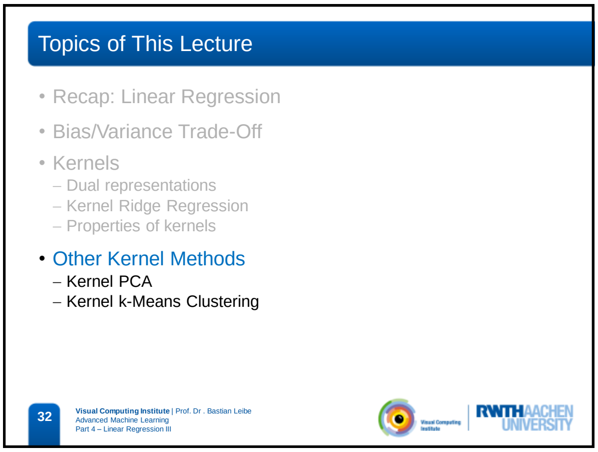
$\Rightarrow$ The resulting algorithm will do the same as the linear version, but in the (hidden) feature space $\mathcal{H}$.

  - Caveat: working in $\mathcal{H}$ is not a guarantee for better performance. A good choice of k and model selection are important!

Slide credit: Christoph Lampert

Outlook

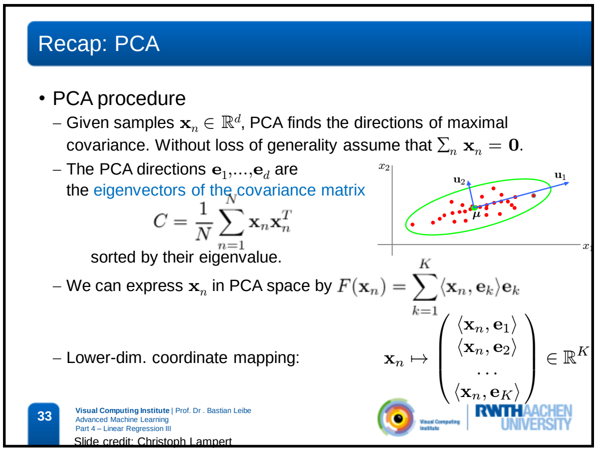
- Kernels are a widely used concept in Machine Learning
  - They are the basis for Support Vector Machines from ML1.
  - We will see several other *kernelized* algorithms in this lecture…
- Examples
  - Gaussian Processes
  - Support Vector Regression
  - Kernel PCA
  - Kernel k-Means
  - …



Topics of This Lecture

- Recap: Linear Regression
- Bias/Variance Trade-Off
- Kernels
  - Dual representations
  - Kernel Ridge Regression
  - Properties of kernels
- Other Kernel Methods
  - Kernel PCA
  - Kernel k-Means Clustering

32



Recap: PCA

- PCA procedure
  - Given samples $\mathbf{x}_n \in \mathbb{R}^d$, PCA finds the directions of maximal covariance. Without loss of generality assume that $\sum_n \mathbf{x}_n = \mathbf{0}$.
  - The PCA directions $\mathbf{e}_1,\ldots,\mathbf{e}_d$ are the eigenvectors of the covariance matrix

$$C = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^T$$

sorted by their eigenvalue.

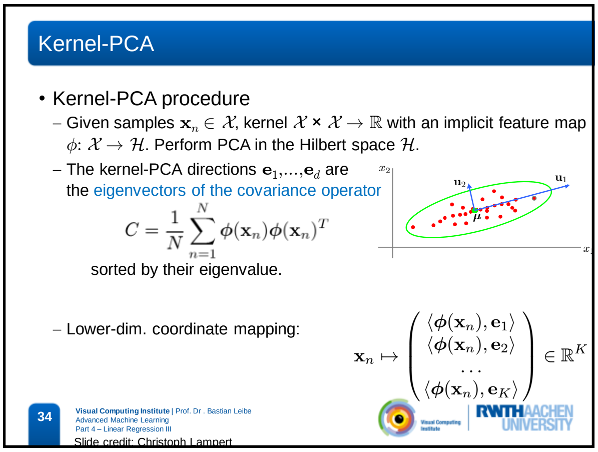
x_2
$\mathbf{u}_2$
$\mathbf{u}_1$
μ
x_1

  - We can express $\mathbf{x}_n$ in PCA space by $F(\mathbf{x}_n) = \sum_{k=1}^{K} \langle \mathbf{x}_n, \mathbf{e}_k \rangle \mathbf{e}_k$
  - Lower-dim. coordinate mapping:

$$\mathbf{x}_n \mapsto \begin{pmatrix} \langle \mathbf{x}_n, \mathbf{e}_1 \rangle \\ \langle \mathbf{x}_n, \mathbf{e}_2 \rangle \\ \ldots \\ \langle \mathbf{x}_n, \mathbf{e}_K \rangle \end{pmatrix} \in \mathbb{R}^K$$

33

Slide credit: Christoph Lampert



Kernel-PCA

- Kernel-PCA procedure
  - Given samples $\mathbf{x}_n \in \mathcal{X}$, kernel $\mathcal{X}$ × $\mathcal{X} \rightarrow \mathbb{R}$ with an implicit feature map ϕ: $\mathcal{X} \rightarrow \mathcal{H}$. Perform PCA in the Hilbert space $\mathcal{H}$.
  - The kernel-PCA directions $\mathbf{e}_1,\ldots,\mathbf{e}_d$ are the eigenvectors of the covariance operator

$$C = \frac{1}{N}\sum_{n=1}^{N} \phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^T$$

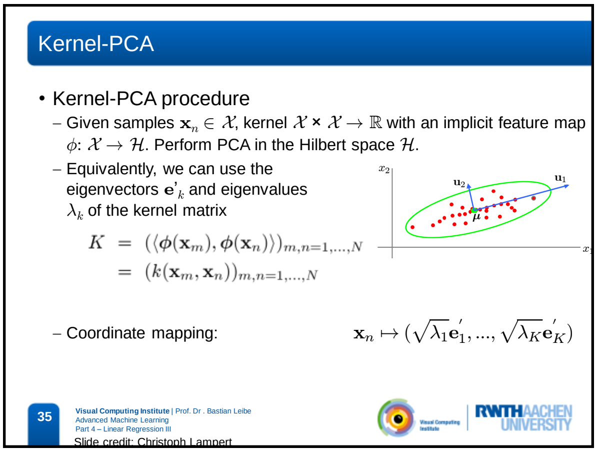
sorted by their eigenvalue.

  - Lower-dim. coordinate mapping:

$$\mathbf{x}_n \mapsto \begin{pmatrix} \langle \phi(\mathbf{x}_n), \mathbf{e}_1 \rangle \\ \langle \phi(\mathbf{x}_n), \mathbf{e}_2 \rangle \\ \ldots \\ \langle \phi(\mathbf{x}_n), \mathbf{e}_K \rangle \end{pmatrix} \in \mathbb{R}^K$$

34

Slide credit: Christoph Lampert



Kernel-PCA

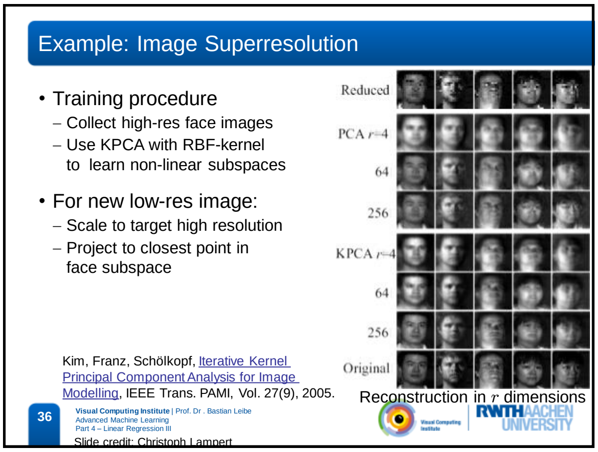
- Kernel-PCA procedure
  - Given samples $\mathbf{x}_n \in \mathcal{X}$, kernel $\mathcal{X}$ × $\mathcal{X} \rightarrow \mathbb{R}$ with an implicit feature map ϕ: $\mathcal{X} \rightarrow \mathcal{H}$. Perform PCA in the Hilbert space $\mathcal{H}$.
  - Equivalently, we can use the eigenvectors $\mathbf{e}'_k$ and eigenvalues λ_k of the kernel matrix

$$\begin{aligned} K &= (\langle \phi(\mathbf{x}_m), \phi(\mathbf{x}_n) \rangle)_{m,n=1,\ldots,N} \\ &= (k(\mathbf{x}_m, \mathbf{x}_n))_{m,n=1,\ldots,N} \end{aligned}$$

  - Coordinate mapping: $\mathbf{x}_n \mapsto (\sqrt{\lambda_1}\mathbf{e}'_1, \ldots, \sqrt{\lambda_K}\mathbf{e}'_K)$

35

Slide credit: Christoph Lampert



Example: Image Superresolution

- Training procedure
  - Collect high-res face images
  - Use KPCA with RBF-kernel to learn non-linear subspaces
- For new low-res image:
  - Scale to target high resolution
  - Project to closest point in face subspace

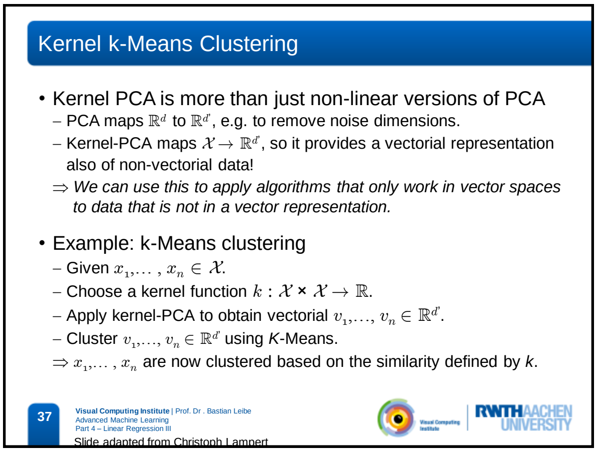
Reduced
PCA $r=4$
64
256
KPCA $r=4$
64
256
Original

Reconstruction in r dimensions

Kim, Franz, Schölkopf, Iterative Kernel Principal Component Analysis for Image Modelling, IEEE Trans. PAMI, Vol. 27(9), 2005.

36

Slide credit: Christoph Lampert



Kernel k-Means Clustering

- Kernel PCA is more than just non-linear versions of PCA
  - PCA maps $\mathbb{R}^d$ to $\mathbb{R}^{d'}$, e.g. to remove noise dimensions.
  - Kernel-PCA maps $\mathcal{X} \rightarrow \mathbb{R}^{d'}$, so it provides a vectorial representation also of non-vectorial data!

  $\Rightarrow$ *We can use this to apply algorithms that only work in vector spaces to data that is not in a vector representation.*
- Example: k-Means clustering
  - Given $x_1,\ldots, x_n \in \mathcal{X}$.
  - Choose a kernel function $k : \mathcal{X}$ × $\mathcal{X} \rightarrow \mathbb{R}$.
  - Apply kernel-PCA to obtain vectorial $v_1,\ldots, v_n \in \mathbb{R}^{d'}$.
  - Cluster $v_1,\ldots, v_n \in \mathbb{R}^{d'}$ using *K*-Means.

  $\Rightarrow$ $x_1,\ldots, x_n$ are now clustered based on the similarity defined by *k*.

37

Slide adapted from Christoph Lampert

Example: Unsupervised Object Categorization

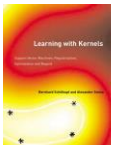
- Automatically group images that show similar objects
  - Represent images by bag-of-word histograms
  - Perform Kernel k-Means Clustering
  - ⇒ Observation: Clusters get better if we use a good image kernel (e.g., χ^2) instead of plain k-Means (linear kernel).

T. Tuytelaars, C. Lampert, M. Blaschko, W. Buntine, Unsupervised object discovery: a comparison, IJCV, 2009.]

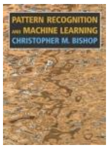
Slide adapted from Christoph Lampert

References and Further Reading

- Kernels are (shortly) described in Chapters 6.1 and 6.4 of Bishop’s book.



Christopher M. Bishop
Pattern Recognition and Machine Learning
Springer, 2006



B. Schölkopf, A. Smola
Learning with Kernels
MIT Press, 2002
http://www.learning-with-kernels.org/

- More information on Kernel PCA can be found in Chapter 12.3 of Bishop’s book. You can also look at Schölkopf & Smola (some chapters available online).