TECHNISCHE
UNIVERSITÄT
DARMSTADT

Statistical Machine Learning

Lecture 10: Linear Dimensionality Reduction & Statistical Learning Theory

Kristian Kersting
TU Darmstadt

Summer Term 2020

Today’s Objectives

- Make you understand how to reduce dimensionality and how to understand the power of function approximators!
- Covered Topics:
  - Principal Component Analysis
  - Statistical Learning Theory

Outline

Outline

Towards Principal Component Analysis

- We have covered a number of linear methods
  - For classification and regression
  - Both are supervised learning settings, i.e., pairs of input/output training data points
- Sometimes, it is quite helpful to analyze the data points themselves
  - Unsupervised learning
  - Particularly: Reduce the dimensionality of the data
  - Possible application: Visualization of the data

Motivation from Linear Least-squares Regression

- In straightforward least-squares linear regression the parameters are computed as

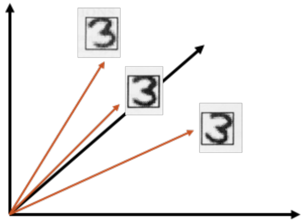
$$\hat{\mathbf{w}} = \left(\hat{\mathbf{X}}\hat{\mathbf{X}}^\intercal\right)^{-1}\hat{\mathbf{X}}\mathbf{y}$$

  where $\hat{\mathbf{X}} \in \mathbb{R}^{d\times n}$ and $\mathbf{y} \in \mathbb{R}^{n\times 1}$

- We need to invert a $d \times d$ matrix, which naively costs $O\left(d^3\right)$
- Hence, it would be helpful to find a new $d^{\text{new}} << d$ to gain computational advantage while not loosing prediction performance

Dimensionality Reduction

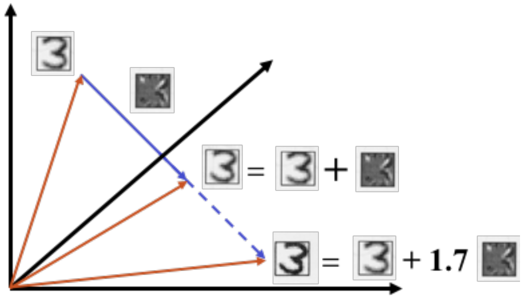
- How can we find more efficient representations for our data? How can we capture the “essence” of the data?
- Example: images of the digit 3



- The images can be represented as points in a high-dimensional space (e.g., with one dimension per pixel, in a 4k image there are around 9 million dimensions!)

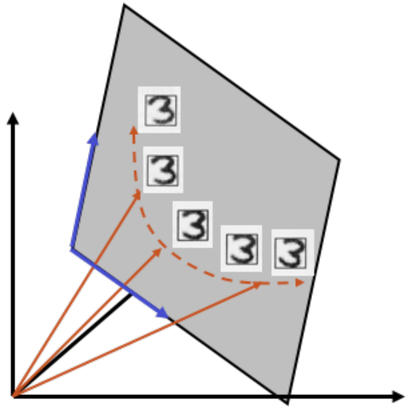
Linear Dimensionality Reduction

- To make things easier, we will once again assume linear models. A data point (here: one image) can be written as a linear combination of bases (here: basis images)



Linear Dimensionality Reduction

- What linear transformations of the data can be used to define a lower-dimensional subspace that captures most of the structure?



Goal of Linear Dimensionality Reduction

- Original data point n: $\boldsymbol{x}^n \in \mathbb{R}^M$
- Low-dimensional representation of data point n: $\boldsymbol{a}^n \in \mathbb{R}^D$ with $D << M$
- Goal: find a mapping

$$\boldsymbol{x}^n \to \boldsymbol{a}^n$$

- Restrict this mapping to be a linear function

$$\boldsymbol{a}^n = \boldsymbol{B}\boldsymbol{x}^n, \quad \boldsymbol{B} \in \mathbb{R}^{D \times M}$$

Simple Observation

- We can always write a vector as

$$\boldsymbol{x} = \sum_{i=1}^{M} a_i \boldsymbol{u}_i, \text{ where } \boldsymbol{u}_i^\top \boldsymbol{u}_j = \delta_{ij}$$

  - $\delta_{ij} = 1$ if $i = j$, 0 otherwise. $\boldsymbol{u}_i$ and $\boldsymbol{u}_j$ are **orthonormal** to each other

- Example

$$\begin{bmatrix} 3 \\ 7 \end{bmatrix} = 3 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 7 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Simple Observation

$$\begin{aligned}
\boldsymbol{x} &= a_1 \boldsymbol{u}_1 + a_2 \boldsymbol{u}_2 \\
a_1 &= \boldsymbol{u}_1^\top (\boldsymbol{x} - a_2 \boldsymbol{u}_2) \\
&= \boldsymbol{u}_1^\top \boldsymbol{x} - \boldsymbol{u}_1^\top a_2 \boldsymbol{u}_2 \\
&= \boldsymbol{u}_1^\top \boldsymbol{x} - a_2 \underbrace{\boldsymbol{u}_1^\top \boldsymbol{u}_2}_{=0} \\
&= \boldsymbol{u}_1^\top \boldsymbol{x}
\end{aligned}$$

- More generally

$$\underbrace{a_i}_{\text{scalar coefficient}} = \underbrace{\boldsymbol{u}_i^\top \boldsymbol{x}}_{\text{projection}}$$

Decomposition

$$\boldsymbol{x}^n = \sum_{i=1}^{D} a_i \boldsymbol{u}_i + \underbrace{\sum_{j=D+1}^{M} b_j \boldsymbol{u}_j}_{\text{error}} \approx \tilde{\boldsymbol{x}}^n$$

- We want the D bases that minimize the mean squared error over the training data

$$\boldsymbol{u}_1, \ldots, \boldsymbol{u}_D = \arg\min_{\boldsymbol{u}_1,\ldots,\boldsymbol{u}_D} E\left(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_D\right) = \arg\min_{\boldsymbol{u}_1,\ldots,\boldsymbol{u}_D} \sum_{n=1}^{N} \left\|\boldsymbol{x}^n - \tilde{\boldsymbol{x}}^n\right\|^2$$

Minimizing the error

- Rewrite the error (assuming a single basis vector)

$$
\begin{aligned}
E(\boldsymbol{u}) &= \sum_{n=1}^{N} \|\boldsymbol{x}^n - \tilde{\boldsymbol{x}}^n\|^2 \\
&= \sum_{n=1}^{N} \|\boldsymbol{x}^n - (\boldsymbol{u}^\intercal \boldsymbol{x}^n)\,\boldsymbol{u}\|^2 \\
&= \sum_{n=1}^{N} \|\boldsymbol{x}^n\|^2 - 2\,(\boldsymbol{u}^\intercal \boldsymbol{x}^n)^2 + (\boldsymbol{u}^\intercal \boldsymbol{x}^n)^2 \cdot \boldsymbol{u}^\intercal \boldsymbol{u} \\
&= \sum_{n=1}^{N} \|\boldsymbol{x}^n\|^2 - (\boldsymbol{u}^\intercal \boldsymbol{x}^n)^2 \\
&= \sum_{n=1}^{N} \|\boldsymbol{x}^n\|^2 - (a_n)^2
\end{aligned}
$$

Minimizing the error

- Rewrite the error (assuming a single basis vector)

$$E(\boldsymbol{u}) \;=\; \sum_{n=1}^{N} \|\boldsymbol{x}^n\|^2 - (\boldsymbol{u}^\intercal \boldsymbol{x}^n)^2 = \sum_{n=1}^{N} \|\boldsymbol{x}^n\|^2 - (a_n)^2$$

- Minimizing the error is equivalent to maximizing the variance of the projection. Assuming a zero mean on the data

$$\max \frac{1}{N} \sum_{n=1}^{N} a_n^2$$

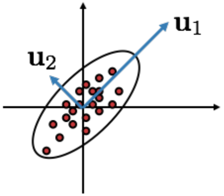
- We can ensure a zero mean projection by subtracting the mean from every data point

$$\boldsymbol{x}^n - \bar{\boldsymbol{x}}$$

Intuition

$$\boldsymbol{x}^n - \bar{\boldsymbol{x}} = \sum_{i=1}^{D} a_i \boldsymbol{u}_i + \sum_{j=D+1}^{M} b_j \boldsymbol{u}_j$$

$$\tilde{\boldsymbol{x}}^n = \sum_{i=1}^{D} a_i \boldsymbol{u}_i + \bar{\boldsymbol{x}}$$

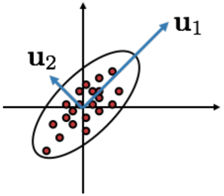


- Projecting onto $\boldsymbol{u}_1$ captures the majority of the variance and hence projecting onto it minimizes the error
- What is that and how do we find the axis of largest variance?

Intuition

$$\boldsymbol{x}^n - \bar{\boldsymbol{x}} = \sum_{i=1}^{D} a_i \boldsymbol{u}_i + \sum_{j=D+1}^{M} b_j \boldsymbol{u}_j$$

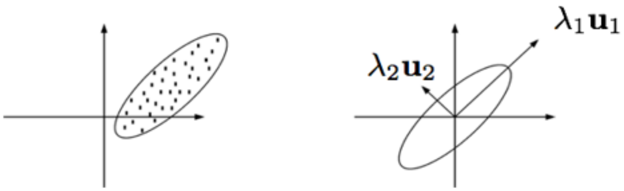
$$\tilde{\boldsymbol{x}}^n = \sum_{i=1}^{D} a_i \boldsymbol{u}_i + \bar{\boldsymbol{x}}$$



- Note that these axes are orthogonal and decorrelate the data; i.e. in the coordinate frame of these axes, the data is uncorrelated (side note: this only works for Gaussians)

Principal Component Analysis

- Goal: find the so-called principal directions, and the variance of the data along each principal direction



- λ_i is the marginal variance along the principal direction $\boldsymbol{u}_i$

Principal Component Analysis

- The first principal direction $\boldsymbol{u}_1$ is the direction along which the variance of the data is maximal

$$\boldsymbol{u}_1 = \arg\max_{\boldsymbol{u}} \boldsymbol{u}^\intercal \boldsymbol{C}\boldsymbol{u} \qquad \text{s.t. } \boldsymbol{u}^\intercal \boldsymbol{u} = 1$$

- The second principal direction maximizes the variance of the data in the orthogonal complement of the first principal direction

Problem Formulation

- Let $\boldsymbol{X} = \left[\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n\right] \in \mathbb{R}^{M \times N}$ be a matrix of N vectors in a M-dimensional input space, with $\boldsymbol{x}^i \in \mathbb{R}^M$
- Let $\boldsymbol{u} \in \mathbb{R}^M$ be a direction (a vector of length 1) in the input space
- The projection of the $j-$th vector $\boldsymbol{x}^j$ onto the vector $\boldsymbol{u}$ can be computed as

$$a_j = \boldsymbol{u}^\intercal \boldsymbol{x}^j = \sum_{i=1}^{M} X_{ij} u_i$$

- The goal is to find a direction $\boldsymbol{u}$ that maximizes the variance of the projections of all input vectors

$$\boldsymbol{x}^j \quad \forall j = 1, \ldots, N$$

Variance of the Projection

$$\bar{a} = \frac{1}{N}\sum_{j=1}^{N} a_j = \frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{M} \mathbf{X}_{ij} u_i = \sum_{i=1}^{M} u_i \mu_i$$

where $\mu_i = \frac{1}{N}\sum_{j=1}^{N} X_{ij}$ and $\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_j$

$$\begin{aligned}
\sigma^2 &= \frac{1}{N}\sum_{j=1}^{N}(a_j - \bar{a}) = \frac{1}{N}\sum_{j=1}^{N}\left(\sum_{i=1}^{M} u_i \mathbf{X}_{ij} - \sum_{i=1}^{M} u_i \mu_i\right)^2 \\
&= \frac{1}{N}\sum_{j=1}^{N}\left(\sum_{i=1}^{M} u_i \hat{\mathbf{X}}_{ij}\right)^2 = \frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{M}\sum_{k=1}^{M} u_i \hat{\mathbf{X}}_{ij} u_k \hat{\mathbf{X}}_{kj} \\
&= \sum_{i=1}^{M}\sum_{k=1}^{M} u_i u_k \underbrace{\frac{1}{N}\left\langle \hat{\boldsymbol{X}}_{i:}, \hat{\boldsymbol{X}}_{k:} \right\rangle}_{\boldsymbol{C}=\frac{1}{N}\hat{\boldsymbol{X}}\hat{\boldsymbol{X}}^\intercal} = \sum_{i=1}^{M}\sum_{k=1}^{M} u_i C_{ik} u_k = \boldsymbol{u}^\intercal \boldsymbol{C}\boldsymbol{u}
\end{aligned}$$

- $\boldsymbol{C}$ is the covariance matrix

TECHNISCHE UNIVERSITÄT DARMSTADT

Maximizing the Variance

- Maximize σ^2 under the constraint that $\|\boldsymbol{u}\| = 1$
- We can translate into an optimization with Lagrangian multipliers

$$F(\boldsymbol{u}; \lambda) = \boldsymbol{u}^\intercal \boldsymbol{C}\boldsymbol{u} - \lambda \left(\sum_{k=1}^{M} u_k^2 - 1 \right) = \sum_{i=1}^{M} \sum_{j=1}^{M} u_i C_{ij} u_j - \lambda \left(\sum_{k=1}^{M} u_k^2 - 1 \right)$$

$$\frac{\partial F}{\partial u_l} = \sum_{j=1}^{M} C_{lj} u_j + \sum_{i=1}^{M} u_i C_{il} - \lambda 2 u_l = 0 \quad \forall l = 1, \ldots, M$$

$$\sum_{j=1}^{M} C_{lj} u_j = \lambda u_l \quad \forall l = 1, \ldots, M \Longleftrightarrow \mathbf{C}\mathbf{u} = \lambda \mathbf{u}$$

Maximizing the Variance

$$\sum_{j=1}^{M} C_{lj} u_j = \lambda u_l \quad \forall l = 1, \ldots, M \Longleftrightarrow \mathbf{Cu} = \lambda \mathbf{u}$$

- Equation of Eigenvalues-Eigenvectors
- The largest Eigenvalue gives us the maximal variance
- The corresponding Eigenvector gives us the direction with maximal variance

TECHNISCHE UNIVERSITÄT DARMSTADT

Eigendecomposition - Reminder

- Let the Eigenvectors and Eigenvalues of $\mathbf{C}$ be $\mathbf{u}_k$ and λ_k for $k \leq M$, i.e., $\mathbf{C}\mathbf{u}_k = \lambda_k \mathbf{u}_k$ with $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_M$
- In matrix form: $\mathbf{C}\mathbf{U} = \mathbf{U}\Lambda$, where $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_M)$ and $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_M]$
- Because $\mathbf{U}$ is orthonormal (we assume that the Eigenvectors have unit norm) we know that $\mathbf{U}\mathbf{U}^\intercal = \mathbf{I}$
- This means that we can decompose $\mathbf{C}$ as

$$\mathbf{C} = \mathbf{U}\Lambda\mathbf{U}^{-1}$$
$$\mathbf{C} = \mathbf{U}\Lambda\mathbf{U}^\intercal$$

Principal Component Analysis

- By definition of a covariance matrix, **C** is **real**, **symmetric** and **positive-definite**. Thus we can decompose it in its Eigendecomposition as

$$\mathbf{C} = \mathbf{U}\Lambda\mathbf{U}^\intercal = \underbrace{\begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_M \end{bmatrix}}_{\text{Eigenvectors}} \underbrace{\begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_2 \end{bmatrix}}_{\text{Eigenvalues}} \begin{bmatrix} \mathbf{u}_1^\intercal \\ \mathbf{u}_2^\intercal \\ \vdots \\ \mathbf{u}_M^\intercal \end{bmatrix}$$

Principal Component Analysis

- **Observation**: If $\lambda_k \approx 0$ for $k > D$ for some $D << M$, then we can use the subset of the first D eigenvectors to define a basis for approximating the data vectors

$$\mathbf{x}^n - \bar{\mathbf{x}} = \sum_{i=1}^{D} a_i \mathbf{u}_i + \sum_{j=D+1}^{M} b_j \mathbf{u}_j$$

$$\mathbf{x}^n \approx \tilde{\mathbf{x}}^n = \bar{\mathbf{x}} + \sum_{i=1}^{D} a_i \mathbf{u}_i \quad \text{where } a_i = \mathbf{u}_i^\intercal \left(\mathbf{x}^n - \bar{\mathbf{x}}\right)$$

- This representation has the minimal mean squared error (MSE) of all linear representations of dimension D

$$\min E\left(\mathbf{u}_1, \ldots, \mathbf{u}_D\right) = \sum_{n=1}^{N} \left\|\mathbf{x}^n - \tilde{\mathbf{x}}^n\right\|^2$$

Principal Component Analysis

- Now we know how we can represent our data in a lower dimensional space in a principled way
  - Center the data around the mean (compute the mean of the data and subtract it)
  - Compute the covariance matrix, decompose it, and choose the first *D* largest Eigenvalues and corresponding Eigenvectors
  - This gives us an (Eigen)basis for representing the data

$$\mathbf{a}^n = \mathbf{B}^\intercal \left(\mathbf{x}^n - \bar{\mathbf{x}}\right)$$
$$\tilde{\mathbf{x}}^n = \bar{\mathbf{x}} + \mathbf{B}\mathbf{a}^n$$

where $\mathbf{B} = \begin{bmatrix} \mathbf{u}_1 & \dots & \mathbf{u}_D \end{bmatrix}$

  - It is also common to normalize the variance of each dimension
- But how to choose D?

Choosing D

- A larger D leads to a better approximation. In the limit, when $D = M$ we stay in the initial data dimensions
- There are at least 2 good possibilities for choosing D

Choosing D

- A larger D leads to a better approximation. In the limit, when $D = M$ we stay in the initial data dimensions
- There are at least 2 good possibilities for choosing D
  1. Choose D based on application performance, i.e. choose the smallest D that makes the application work *well enough*

Choosing D

- A larger D leads to a better approximation. In the limit, when $D = M$ we stay in the initial data dimensions
- There are at least 2 good possibilities for choosing D
  1. Choose D based on application performance, i.e. choose the smallest D that makes the application work *well enough*
  2. Choose D so that the eigenbasis captures some fraction of the variance (for example $\eta = 0.9$)
     The eigenvalue λ_i describes the marginal variance captured by $\mathbf{u}_i$

Choose D s.t. $\sum_{i=1}^{D} \lambda_i \geq \eta \sum_{i=1}^{M} \lambda_i$

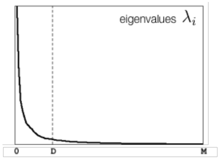


Image Representation

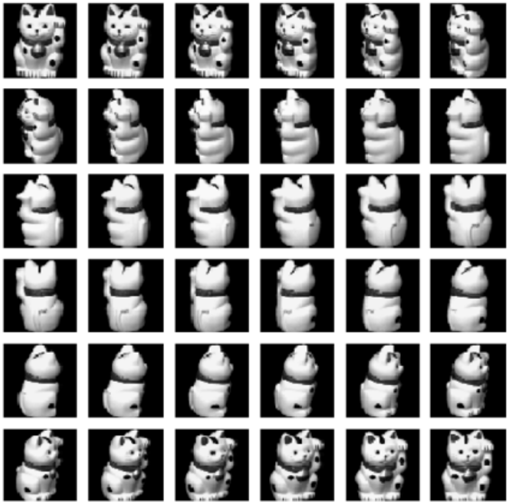
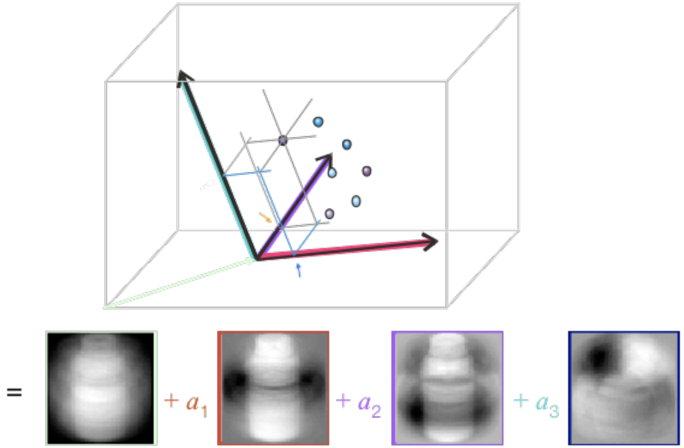


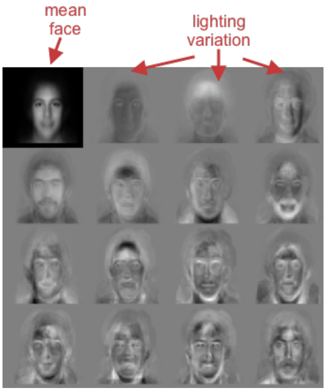
Image Representation with PCA



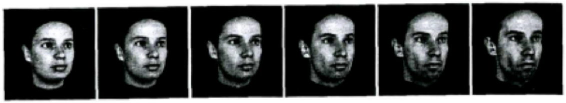
EigenFaces

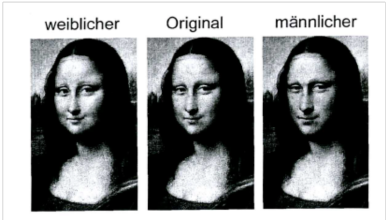
- The first popular use of PCA for object recognition was for the detection and recognition of faces [Turk and Pentland, 1991]

- Collect a face ensemble
- Normalize for contrast, scale, & orientation
- Remove backgrounds
- Apply PCA & choose the first *D* eigen-images that account for most of the variance of the data

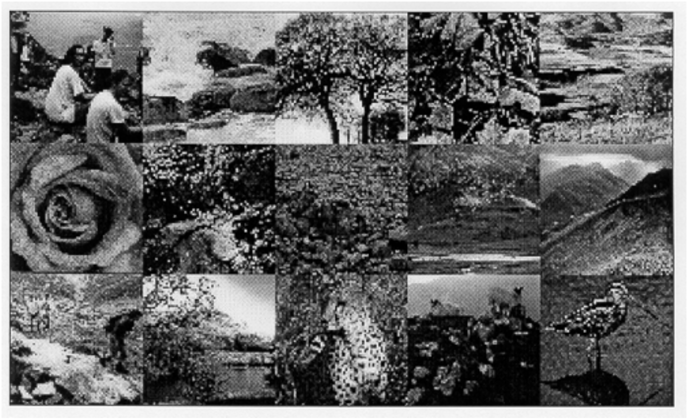


Morphing with EigenFaces





Generic Image Ensembles

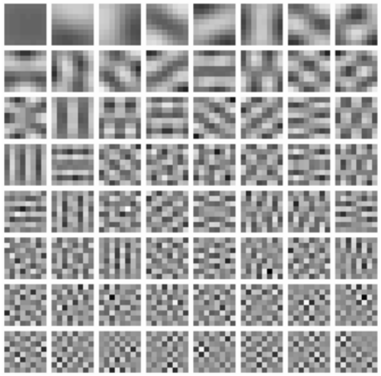


Generic Image Ensembles

Is there a low dimensional model describing natural images?

PCA of Natural Image Patches

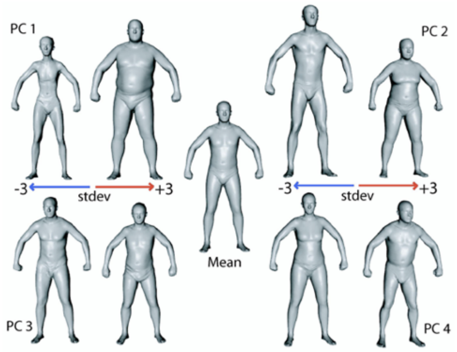
8x8 image patches



What do these bases look like?

PCA Model of Body Shapes

PCA on a detailed triangle model of human bodies [Anguelov et al. 05]



PCA Summary and Applications

- **Summary**
  - PCA projects the data into a linear subspace
  - PCA maximizes the variance of the projection
  - PCA minimizes the error of the reconstruction
- **Applications**
  - PCA allows us to transform a high-dimensional input space to a low-dimensional feature space, while capturing the essence of the data
  - PCA finds a more *natural* coordinate system for the data
  - PCA is a very common preprocessing step for high-dimensional input data

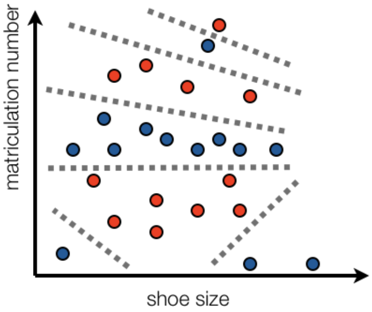
Outline

Our approach to Machine Learning until now

- We have mostly dealt with *Classical Statistical Learning*
  - Estimate a parameter vector **w** of a fixed model (*learning machine*)
  - Examples
    - Maximum Likelihood estimation: determine the parameters that model a density function
    - Discriminant functions: determine the orientation of the separating hyperplane
- Learning occurs by optimizing the model parameters
- In this setting we assume that we already know in advance the correct model (Bayesian learning is the exception)
  - For instance we model the density function with a linear combination of features, a neural network, a Gaussian, ...

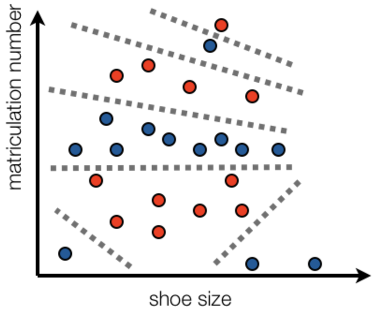
Generalization Abilities

- Example: Classify students who will pass the exam. What are some common challenges?



Generalization Abilities

- Example: Classify students who will pass the exam. What are some common challenges?



- Problem 1: Possibly irrelevant features
- Problem 2: Overfitting from overly complex model
- We are really interested in the generalization ability and the corresponding risk

Statistical Learning Theory

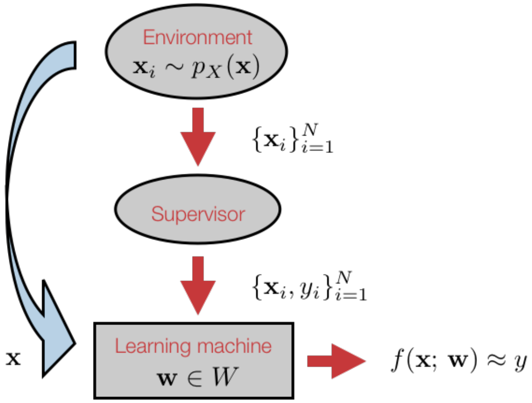
- Statistical Learning Theory
  - Does not assume that the correct model is known in advance
  - The goal is to choose an *optimal* model from a set of specified models
  - It is a form of *model selection*
- Optimality here means
  - Ability of the model to generalize, i.e. to have the lowest error probability on all data, and not only on the training data

Statistical Learning Theory

Vladimir Vapnik [1936-]

- Statistical Learning Theory (from Vladimir Vapnik)
  - Is concerned with the question how one can control the generalization abilities of a learning machine
  - Aims at a formal description of the generalization ability
  - The goal is to develop a rigorous theory as opposed to commonly used heuristics
- We will see later that this may be a noble goal, but the theory unfortunately does not say that much about real problems...

Supervised Learning


Environment
$\mathbf{x}_i \sim p_X(\mathbf{x})$
$\{\mathbf{x}_i\}_{i=1}^N$
Supervisor
$\{\mathbf{x}_i, y_i\}_{i=1}^N$
$\mathbf{x}$
Learning machine
$\mathbf{w} \in W$
$f(\mathbf{x};\ \mathbf{w}) \approx y$

Statistical Learning Theory

- Supervised learning as an example
  - The environment is stationary, i.e., the data points $\mathbf{x}_i$ have an unknown, but fixed probability density

$$\mathbf{x}_i \sim p_X$$

  - The supervisor returns the intended classification label y for every data point $\mathbf{x}$, possibly with some added noise υ

$$y = g(\mathbf{x}, \upsilon)$$

  - The learning machine is represented through a class of functions (with parameters $\mathbf{w}$) that return an output y for every input $\mathbf{x}$

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

Statistical Learning Theory

- Supervised learning from the learning machine view
  - Choose a particular function

$$y = f\left(\mathbf{x}, \mathbf{w}\right)$$

  - Given a set of training examples

$$\{\mathbf{x}_i, y_i\}_{i=1}^{N}$$

  - Goal: the desired output y should be approximated *optimally*
- Assessment of *optimality*
  - A loss function, for example the quadratic loss

$$L\left(y, f\left(\mathbf{x}, \mathbf{w}\right)\right) = \left(y - f\left(\mathbf{x}, \mathbf{w}\right)\right)^2$$

Assessment of optimality - Risk

- Loss function

$$L\left(y, f\left(\mathbf{x}, \mathbf{w}\right)\right)$$

- Empirical risk

$$R_{\text{emp}}\left(\mathbf{w}\right) = \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, f\left(\mathbf{x}_i, \mathbf{w}\right)\right)$$

  where N is the number of data points

- Example with the quadratic loss function

$$R_{\text{emp}}\left(\mathbf{w}\right) = \frac{1}{N}\sum_{i=1}^{N} \left(y_i - f\left(\mathbf{x}_i, \mathbf{w}\right)\right)^2$$

TECHNISCHE UNIVERSITÄT DARMSTADT

Risk

- In reality, we are instead interested in the True Risk

$$
\begin{aligned}
R(\mathbf{w}) &= \int L(y, f(\mathbf{x}, \mathbf{w}))\, p(\mathbf{x}, y)\, d\mathbf{x}dy \\
&= \mathbb{E}_{\mathbf{x}, y \sim p(\mathbf{x}, y)}\left[L(y, f(\mathbf{x}, \mathbf{w}))\right]
\end{aligned}
$$

  where $p(\mathbf{x}, y)$ is the true joint probability density of $\mathbf{x}$ and y
- The risk is the expected error over all data sets
- The risk is the expectation of the generalization error
- What is the problem here?

Risk

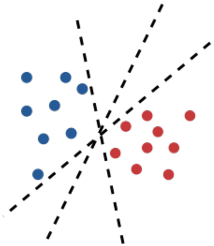
- In reality, we are instead interested in the True Risk

$$R(\mathbf{w}) = \int L(y, f(\mathbf{x}, \mathbf{w}))\, p(\mathbf{x}, y)\, d\mathbf{x}dy$$
$$= \mathbb{E}_{\mathbf{x},y\sim p(\mathbf{x},y)}\left[L(y, f(\mathbf{x}, \mathbf{w}))\right]$$

  where $p(\mathbf{x}, y)$ is the true joint probability density of $\mathbf{x}$ and y

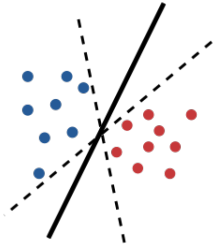
- The risk is the expected error over all data sets
- The risk is the expectation of the generalization error
- What is the problem here?
  - $p(\mathbf{x}, y)$ is fixed but usually unknown
  - We cannot compute the (actual) risk directly

Empirical vs True Risk



- All 3 decision boundaries have zero empirical risk
- Which one is preferable?
- Which one generalizes best?

Empirical vs True Risk



- All 3 decision boundaries have zero empirical risk
- Which one is preferable?
- Which one generalizes best?

Empirical vs True Risk

$$R(\mathbf{w}) = \mathbb{E}_{\mathbf{x},y\sim p(\mathbf{x},y)}\left[L\left(y, f\left(\mathbf{x},\mathbf{w}\right)\right)\right], \quad R_{\text{emp}}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, f\left(\mathbf{x}_i,\mathbf{w}\right)\right)$$

- True risk
  - Advantage: Measure for the generalization ability
  - Disadvantage: In general, we do not know $p(\mathbf{x}, y)$
- Empirical risk
  - Disadvantage: No direct measure for the generalization ability
  - Advantage: Does not depend on $p(\mathbf{x}, y)$
  - Learning algorithms often minimize the empirical risk
- We are interested in the dependencies between these two risks

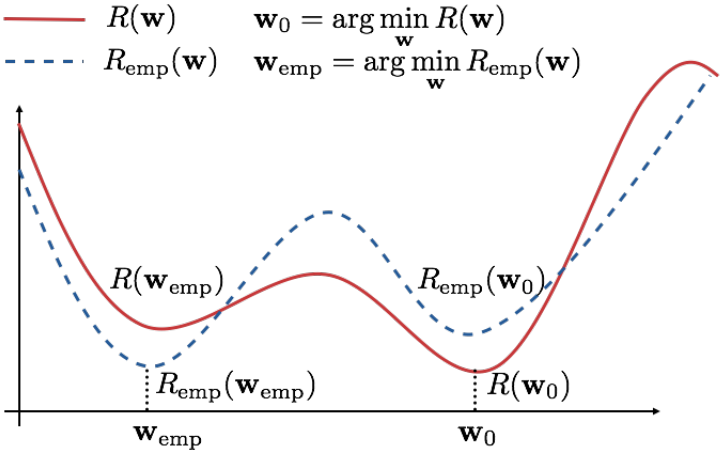
Empirical vs True Risk

- You may be asking yourself: we learned that with Monte Carlo we can estimate an expectation by sampling and averaging. So, isn't the empirical risk the same as the true risk?

$$R(\mathbf{w}) = \mathbb{E}_{\mathbf{x},y\sim p(\mathbf{x},y)}\left[L\left(y, f\left(\mathbf{x},\mathbf{w}\right)\right)\right] \approx \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, f\left(\mathbf{x}_i,\mathbf{w}\right)\right) = R_{\mathrm{emp}}(\mathbf{w})$$

- It is just an approximation, which works very well if our distribution is very concentrated
- The approximation is very good if we have infinitely many samples from the joint distribution
- We will see now how the two risks relate

Convergence Properties



Convergence Properties - Derivation

- We assume that the empirical risk converges to the true risk as we supply more and more training data

$$\inf_{\mathbf{w}} R_{\text{emp}}(\mathbf{w}) \xrightarrow{P} \inf_{\mathbf{w}} R(\mathbf{w}) \quad \text{as } N \to \infty$$

where inf stands for *infimum*

Convergence Properties - Derivation

- We assume that the empirical risk converges to the true risk as we supply more and more training data

$$\inf_{\mathbf{w}} R_{\text{emp}}(\mathbf{w}) \xrightarrow{P} \inf_{\mathbf{w}} R(\mathbf{w}) \quad \text{as } N \to \infty$$

  where inf stands for *infimum*

- We further assume that the convergence has to be uniform

$$P\left(\sup_{\mathbf{w}} \left|R(\mathbf{w}) - R_{\text{emp}}(\mathbf{w})\right| > \epsilon\right) = 0 \quad \text{as } N \to \infty$$

  where sup stands for *supremum*

- Intuitively, "the learning machine gets better as we give it more training data"

Convergence Properties - Derivation

- If the convergence is uniform

$$P\left(\sup_{\mathbf{w}} \left|R\left(\mathbf{w}\right) - R_{\text{emp}}\left(\mathbf{w}\right)\right| > \epsilon\right) < p^{\star} \quad \text{for some } p^{\star} > 0$$

- Then with probability $(1 - p^{\star})$ it holds that

$$\begin{aligned} \left|R\left(\mathbf{w}_{\text{emp}}\right) - R_{\text{emp}}\left(\mathbf{w}_{\text{emp}}\right)\right| &< \epsilon \\ \left|R\left(\mathbf{w}_{0}\right) - R_{\text{emp}}\left(\mathbf{w}_{0}\right)\right| &< \epsilon \end{aligned}$$

- Hence we have

$$P\left(\left|R\left(\mathbf{w}_{0}\right) - R_{\text{emp}}\left(\mathbf{w}_{\text{emp}}\right)\right| > 2\epsilon\right) < p^{\star}$$

Empirical Risk Minimizing Principle

- Necessary and sufficient condition: uniform convergence

$$P\left(\sup_{\mathbf{w}} \left|R\left(\mathbf{w}\right) - R_{\text{emp}}\left(\mathbf{w}\right)\right| > \epsilon\right) = 0 \quad \text{as } N \to \infty$$

- Compute the empirical risk from the available training data $\{\mathbf{x}_i, y_i\}_{i=1}^{N}$

$$R_{\text{emp}} = \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, f\left(\mathbf{x}_i, \mathbf{w}\right)\right)$$

- Minimizing the empirical risk guarantees that we minimize the true risk in the limit of $N \to \infty$

Empirical Risk Minimizing Principle

- Advantages
  - We have a formal criterion as to what we can expect in terms of generalization
  - Necessary and sufficient condition is the uniform convergence
- Disadvantages
  - In reality, the amount of training data is very limited
  - We can't easily "take the limit" of $N \to \infty$
  - Nonetheless, what we have essentially been assuming so far in many cases

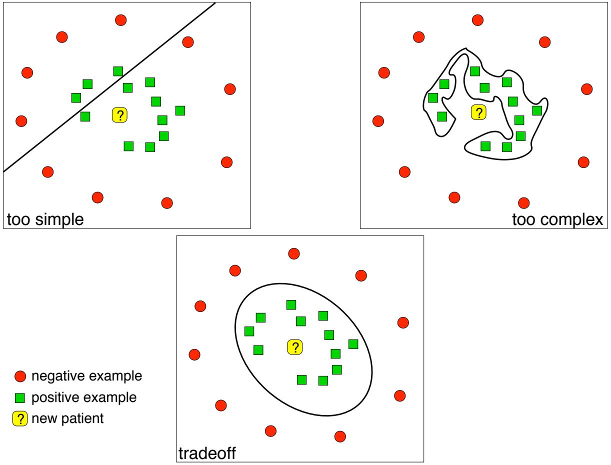
Risk Bound

- Idea: determine an upper bound on the true risk based on the empirical risk

$$R(\mathbf{w}) \leq R_{\text{emp}}(\mathbf{w}) + \epsilon(N, p^{\star}, h)$$

  - N is the number of training examples
  - $p^{\star}$ is the probability that the bound is met
  - h is the *learning power* of the learning machine, formally called the VC-dimension

Learning Power



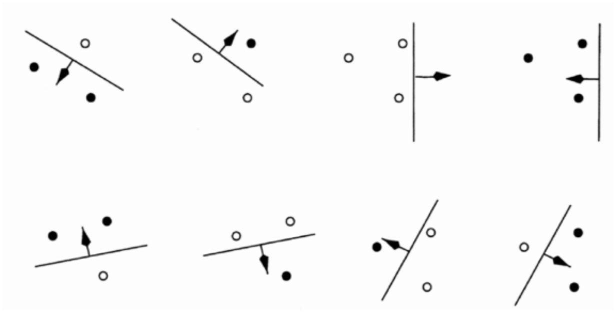
[Florian Markowetz]

VC-Dimension

- VC stands for Vapnik-Chervonenkis
- Informal definition of the VC-dimension
  - The VC-dimension of a family of functions is the maximal number of data points that can be correctly classified by a function from that family (no matter which label configuration the data points have)
  - The VC-dimension is a measure for the capacity, or “learning power”, of a classifier

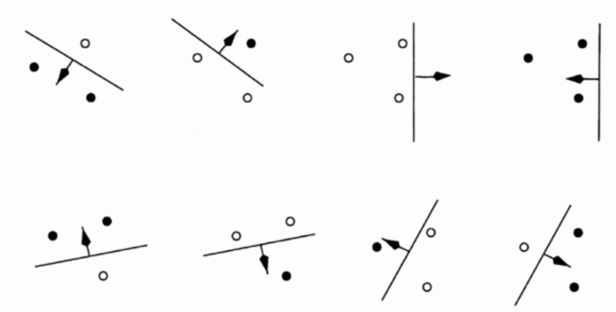
VC-Dimension Example

- The VC-dimension of linear classifiers (lines) in $\mathbb{R}^2$ is 3



VC-Dimension Example

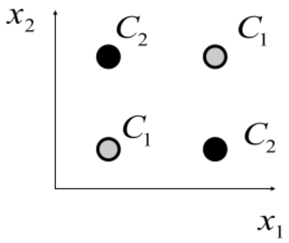
- The VC-dimension of linear classifiers (lines) in $\mathbb{R}^2$ is 3



- More generally, the VC-dimension of linear classifiers (hyperplanes) in $\mathbb{R}^n$ is $n+1$.

VC-Dimension Example

- The VC-dimension of linear classifiers (lines) in $\mathbb{R}^2$ is 3. Why is it not bigger than 3?



VC-Dimension

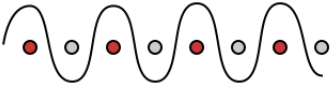
- Intuition (but unfortunately not quite right)
  - The VC-dimension is directly related to the number of parameters
- Counter example

$$\begin{aligned} f(x, \mathbf{w}) &= g(\sin(w_1 x + w_0)) \\ g(x) &= \begin{cases} 1 & x > 0 \\ -1 & x \leq 0 \end{cases} \end{aligned}$$

  - Has only two parameters but infinite VC-dimension

VC-Dimension Counter Example

$$
\begin{aligned}
f(x, \mathbf{w}) &= g(\sin(w_1 x + w_0)) \\
g(x) &= \begin{cases} 1 & x > 0 \\ -1 & x \leq 0 \end{cases}
\end{aligned}
$$



- For other patterns of data, there is always a setting that correctly classifies any labeling of a fixed number of points

Risk Bound Example

- Loss function

$$L(y, f(\mathbf{x}, \mathbf{w})) = \frac{1}{2} |y - f(\mathbf{x}, \mathbf{w})|$$

- True risk

$$R(\mathbf{w}) = \int \frac{1}{2} |y - f(\mathbf{x}, \mathbf{w})| \, p(\mathbf{x}, y) \, \mathrm{d}\mathbf{x}\mathrm{d}y$$

- Empirical risk

$$R_{\text{emp}}(\mathbf{w}) = \frac{1}{2N} \sum_{i=1}^{N} |y_i - f(\mathbf{x}_i, \mathbf{w})|$$

Risk Bound Example

- With probability $p^\star$ it holds that

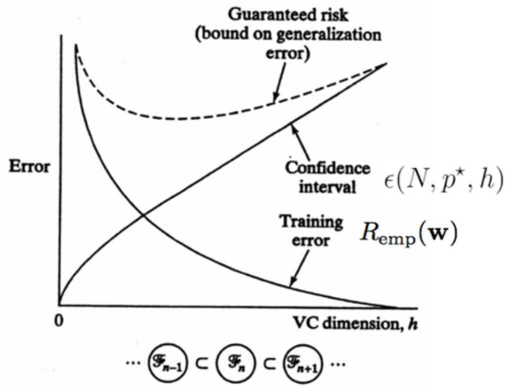
$$R(\mathbf{w}) \leq R_{\text{emp}}(\mathbf{w}) + \sqrt{\frac{h\left(\log\left(2N/h\right)+1\right) - \log\left(\left(1-p^\star\right)/4\right)}{N}}$$

- The upper bound is independent of $p(\mathbf{x}, y)$
- Typically we cannot compute the true risk
- But if we know the VC-dimension, we can compute a bound of the type

$$R(\mathbf{w}) \leq R_{\text{emp}}(\mathbf{w}) + \epsilon\left(N, p^\star, h\right)$$

- Note, however, that in practice this bound is very loose

Risk Bound


Guaranteed risk (bound on generalization error)
Error
Confidence interval
$\epsilon(N, p^\star, h)$
Training error
$R_{\text{emp}}(\mathbf{w})$
0
VC dimension, h
$\cdots \mathcal{F}_{n-1} \subset \mathcal{F}_n \subset \mathcal{F}_{n+1} \cdots$

Structural Risk Minimization

- Given a family of n models $f_i\left(\mathbf{x}_i; \mathbf{w}_i\right)$ with non-decreasing VC-dimension
$$h_1 \leq h_2 \leq \ldots \leq h_n$$
- Minimize the empirical risk for every model
- Choose the model that minimizes the risk bound (Right Hand Side of the true/empirical risk inequality)
- In general, this is not the same model that minimizes the empirical risk
- This is formally justified by the bound on the true risk
- The result is only sensible, however, if the upper bound on the true risk is a tight bound

Outline

3. Wrap-Up

You know now:

- What dimensionality reduction is and why do we need it
- What the intuition behind PCA is
- Why maximizing the variance of the projection is a good idea
- How PCA relates to Eigenvectors and Eigenvalues
- What Statistical Learning Theory is
- What empirical and true risk are and how they relate
- Why looking just at the empirical risk is not enough
- What the VC-Dimension is
- How to relate the VC-Dimension with the model complexity

Self-Test Questions

- What does dimensionality reduction mean?
- What is PCA? What are the three things that it does?
- What are the roles of the Eigenvectors and Eigenvalues in PCA?
- Can you describe applications of PCA?
- What does risk in statistical learning theory mean?
- How is the true risk different from the empirical risk?
- What is the learning power of a function approximator?
- What is expressed by a VC-Dimension?
- Is the VC-Dimension always correlated with the number of parameters?

Homework

- Reading Assignment for next lecture
  - Bishop 5.1, 5.2, 5.3