Applications of & Introduction to Artificial Intelligence

Support Vector Machines for Image Recognition

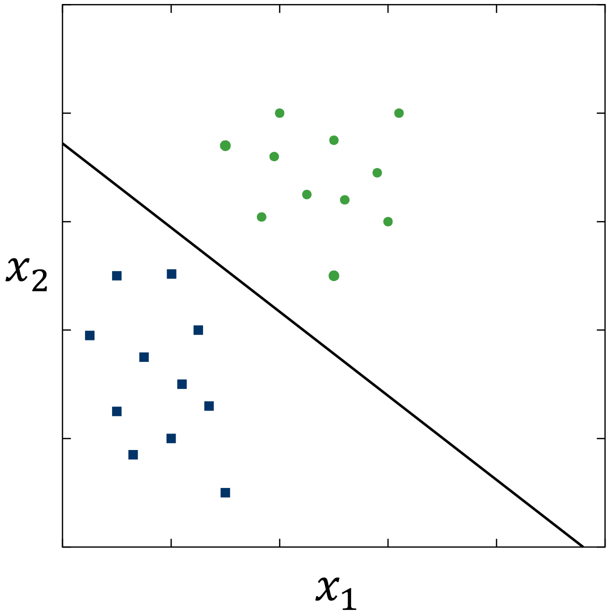
Technische Hochschule Rosenheim

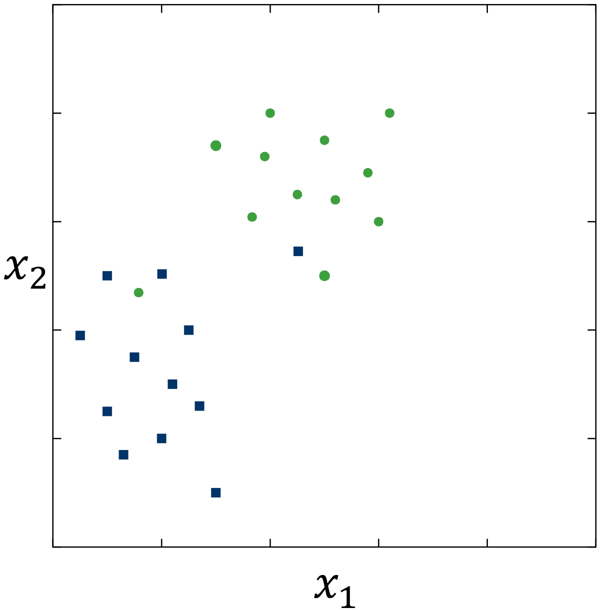
Sommer 2020

Prof. Dr. J. Schmidt

Motivation

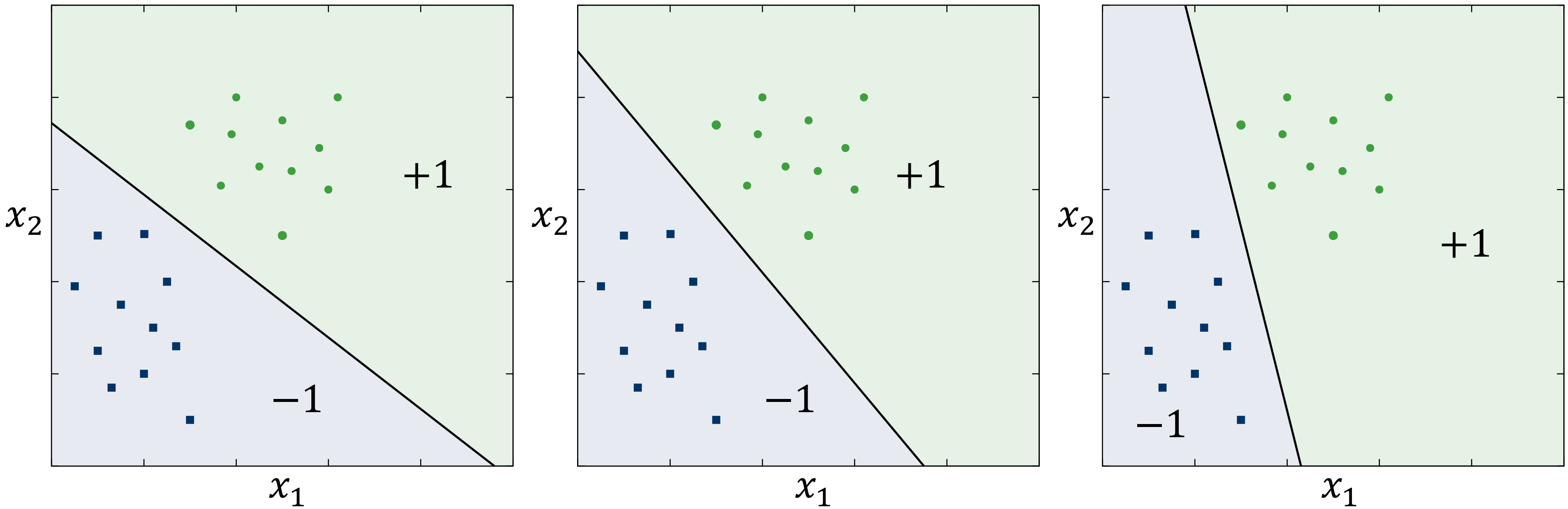
- assume two linearly separable classes
- compute linear decision boundary that
  - allows for separation of training data
  - generalizes well





Motivation

Many, many solutions…



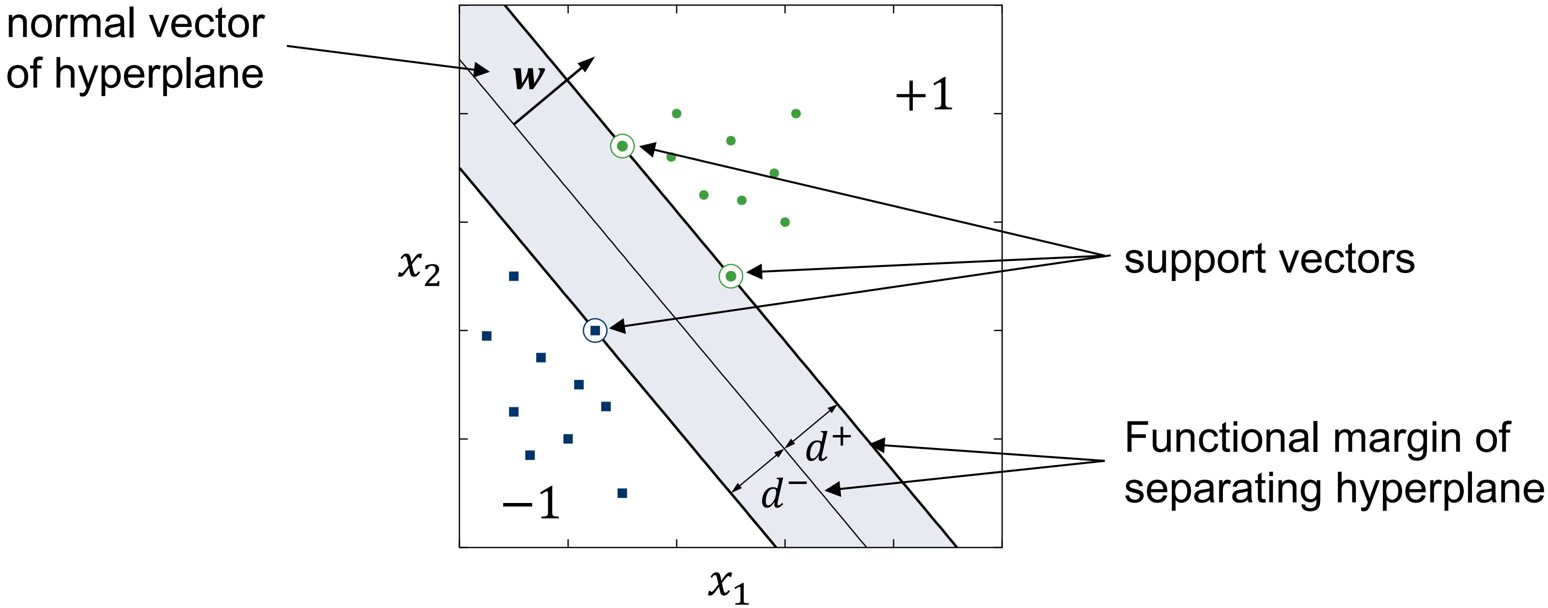
Optimal Separating Hyperplane

Vapnik 1996: Optimal separating hyperplane that

- separates two classes and
- maximizes the distance to the closest point from either class.

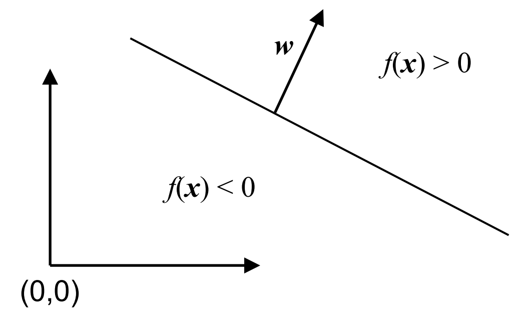
This results in

- unique solution for hyperplanes, and
- (in most cases) better generalization.



Optimal Separating Hyperplane

- Plane equation: $f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0$
  - normal vector: $\boldsymbol{w}$
  - point on plane: $f(\boldsymbol{x}) = 0$
  - point above plane: $f(\boldsymbol{x}) > 0$
  - point below plane: $f(\boldsymbol{x}) < 0$
  - ”above” = in direction of plane normal



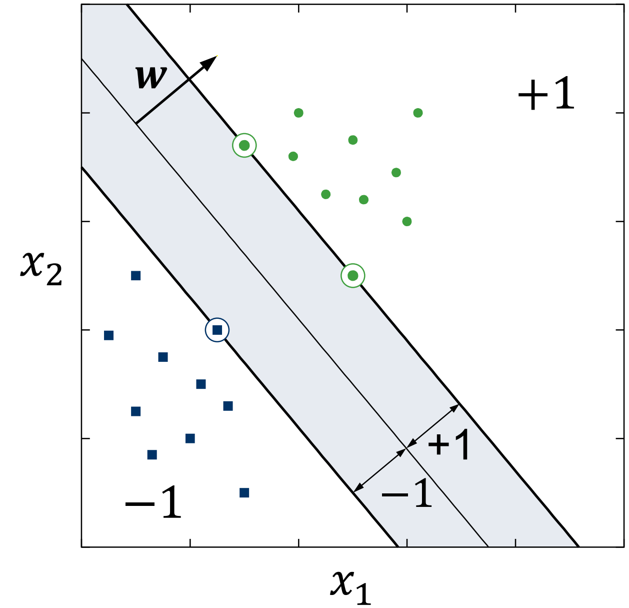
- Signed distance d of a point to hyperplane
  - normalize $\boldsymbol{w}$, such that $|\boldsymbol{w}| = 1$:

$$d = p(\boldsymbol{x}) = \frac{1}{|\boldsymbol{w}|} f(\boldsymbol{x}) = \frac{1}{|\boldsymbol{w}|} \boldsymbol{w}^T \boldsymbol{x} + \frac{1}{|\boldsymbol{w}|} w_0$$

  - distance of plane from origin: $-\frac{1}{|\boldsymbol{w}|} w_0$

SVM – Classification

- data point: $\boldsymbol{x}_i$
- class of data point $\boldsymbol{x}_i$ is $y_i \in \{-1, +1\}$
- Classifier: $g(\boldsymbol{x}_i) = \operatorname{sgn}(\boldsymbol{w}^T\boldsymbol{x}_i + w_0)$



- Functional margin of $\boldsymbol{x}_i$: $y_i\,(\boldsymbol{w}^T\boldsymbol{x}_i + w_0)$
  - can be increased/decreased by scaling plane equation
  - $\rightarrow$ scale such that support vectors have distance $-1/+1$
- Functional margin for data set:
  2x minimum functional margin of all points: $\frac{2}{|\boldsymbol{w}|}$

SVM – Training

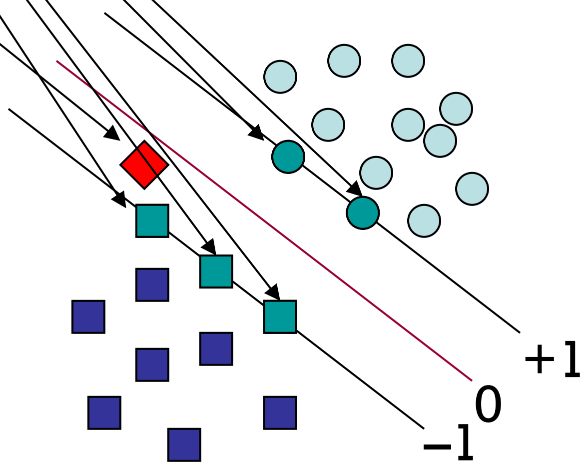
- Training =
  - find hyperplane maximizing the margin $\frac{2}{|\boldsymbol{w}|}$
    - subject to constraint $y_i\ (\boldsymbol{w}^T \boldsymbol{x}_i + w_0) \geq 1$ for all data points
  - instead of plane equation, only support vectors $\boldsymbol{x}_i$ and their corresponding Lagrange multipliers λ_i are required
- Remarks
  - Details of training algorithm are not discussed here
  - this is a convex optimization problem
    - local optimum is always a global one – solution is unique
  - there exist efficient algorithms for convex optimization

SVM – Classification with threshold

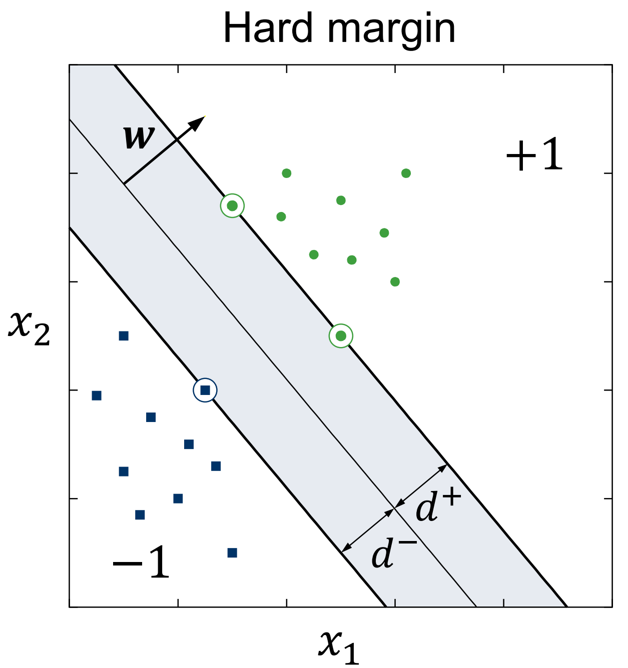
Classification: $g(\boldsymbol{x}) = \sum_i \lambda_i y_i \boldsymbol{x}_i^T \boldsymbol{x} + w_0$

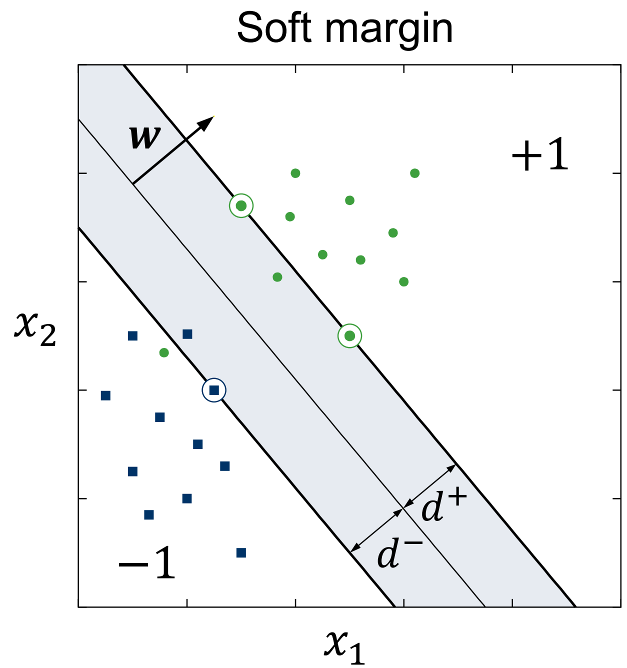
inner (scalar) product

- Classification without threshold
  - decide for class based on $g(\boldsymbol{x}) < 0$ or $g(\boldsymbol{x}) > 0$
- Classification with confidence threshold t
  - $g(\boldsymbol{x}) < -t$: class -1
  - $g(\boldsymbol{x}) > t$: class $+1$
  - $-t < g(\boldsymbol{x}) < t$: reject



Hard and Soft Margin Problem



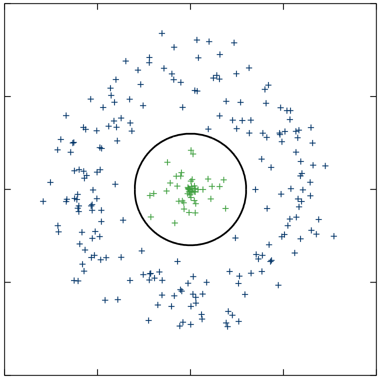


data are not linearly separable in this case

- allow some errors
- allow miss-classification of difficult or noisy samples

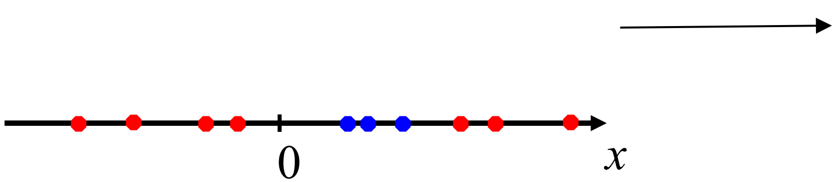
Kernels / Non-linear Boundaries

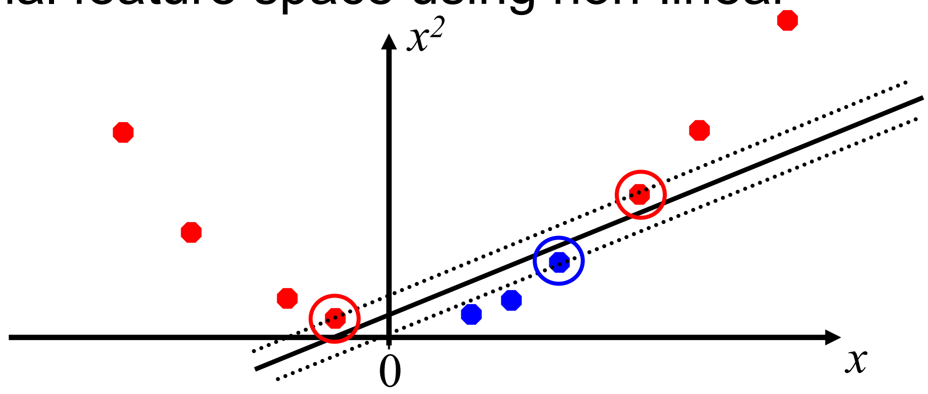
- Limitations of linear decision boundaries
  - too simple for most practical purposes
  - non-linearly separable data cannot be classified
  - noisy data cause problems



?

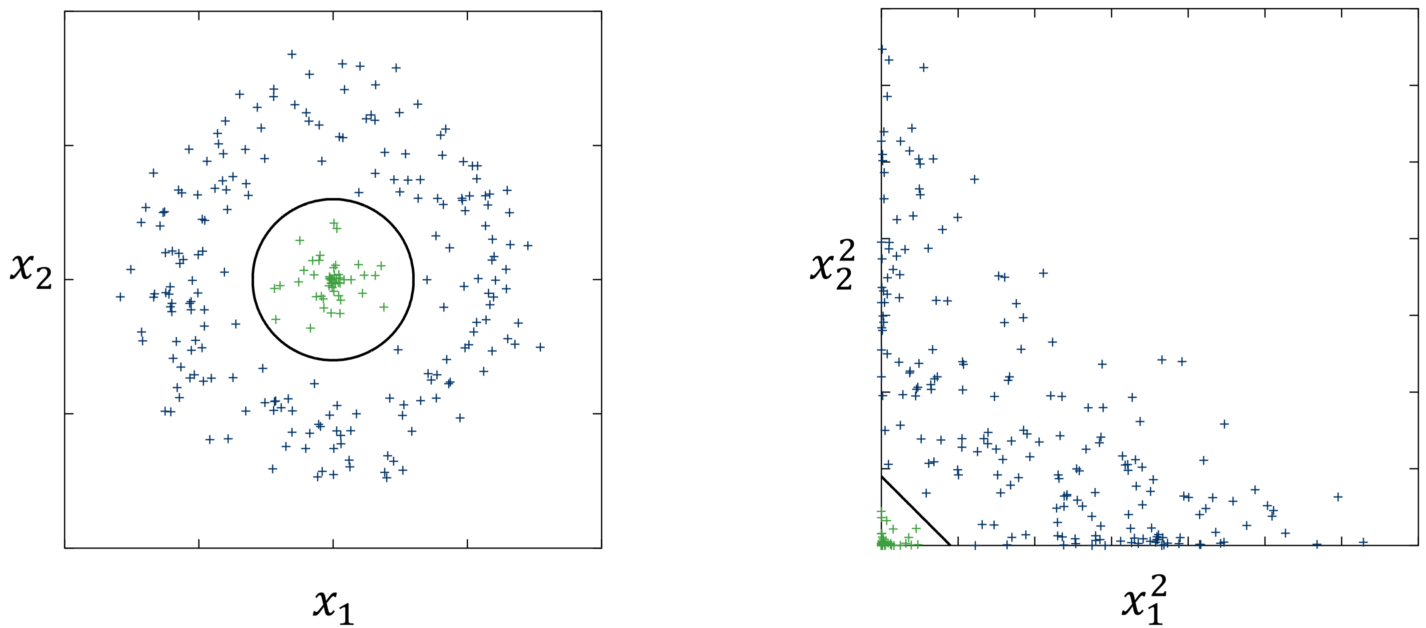
- Possible solution
  - Map data to higher dimensional feature space using non-linear feature transform,
  - then use a linear classifier





Feature Transforms

Select a feature transform $\phi\colon \mathbb{R}^d \to \mathbb{R}^D$ such that the resulting features $\phi(\boldsymbol{x}_i)$ are linearly separable.



Applied feature transform in example: $\phi(\boldsymbol{x}_i) = (x_1^2, x_2^2)^T$

Kernel-Trick

The feature transforms can be easily incorporated into SVMs:
Replace $\boldsymbol{x}_i^T \boldsymbol{x}$ by $\phi^T(\boldsymbol{x}_i)\phi(\boldsymbol{x}) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}) \rangle$ $\langle \cdot \rangle$ notation for inner product

Classification/Decision boundary:

$$g(\boldsymbol{x}) = \sum_i \lambda_i y_i \phi^T(\boldsymbol{x}_i)\phi(\boldsymbol{x}) + w_0 = \sum_i \lambda_i y_i \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}) \rangle + w_0$$

- in SVM training/classification, data appear only in the form of inner products $\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$
- a Kernel-function is a function computing this inner product directly:
  $$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$$
  - i.e., without first transforming the features using $\phi(\boldsymbol{x})$
  - it can be computed in the original low-dimensional space!

Common Kernel Functions

- Linear: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$
- Polynomial: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + 1\right)^d$
- Laplacian radial basis function (RBF):

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\frac{\left\|x_i - x_j\right\|_1}{\sigma^2}}$$

- Gaussian radial basis function (RBF):

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\frac{\left\|x_i - x_j\right\|_2^2}{\sigma^2}}$$

- Sigmoid: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh\left(\alpha \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + \beta\right)$

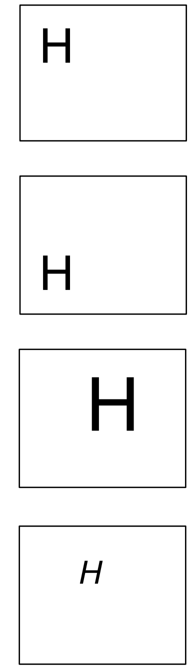
Multiclass-SVM

- split into multiple binary classifications
- one-vs-all
  - one binary SVM per class, separating this class from all others
  - winner-takes all strategy (winner = class with highest value)
- one-vs-one
  - train binary SVMs for each pair of classes
  - each SVM votes: max-wins strategy

- SVMs in scikit-learn:
  https://scikit-learn.org/stable/modules/svm.html

Features for Image Recognition

- Pre-Processing (depending on input and application)
  - conversion to gray-scale
  - reduce noise (Median, low-pass filter)
  - compute edge images (Sobel, Laplace, Canny)
  - segmentation of relevant objects
  - resize/crop images (all of same size)



- Normalization (only parameters irrelevant to class!)
  - position and/or orientation of relevant objects in image
  - size of objects in image
  - illumination
    (e.g. same mean gray value/variance)
  - subtract mean image of training data set

Features for Image Recognition

- feed in (pre-processed) image pixels
  - convert to vector (row-wise or column-wise) – 2D neighborhood information is lost
  - pre-processing: at least subtract mean image vector of training set
- compute features from image
  - and collect these in a feature vector
    - more is not necessarily better!
    - apply feature normalization if necessary (e.g. z-Score)
    - many possibilities
  - example: use first n coefficients of orthogonal transformation
    - Discrete Fourier Transform (DFT)
    - Discrete Cosine Transform (DCT)
    - Principal Component Analysis (PCA)
    - Discrete Wavelet-Transforms (DWT)

Discrete Cosine Transform (DCT)

Computation of a 1D DCT for N-dimensional input vector $\boldsymbol{f}$

$$\boldsymbol{c} = \boldsymbol{\Phi f}$$

with $\Phi_{jk} = \sqrt{\frac{2}{N}} \cos\left(\frac{\pi}{N}\left(j + \frac{1}{2}\right)\left(k + \frac{1}{2}\right)\right)$

DCT is separable, i.e., for an image:
1. transform column vectors
2. transform transformed rows

(or vice versa)

- Matrix $\boldsymbol{\Phi}$
  - is square (size defined by input vector $\boldsymbol{f}$)
  - is orthogonal, i.e.

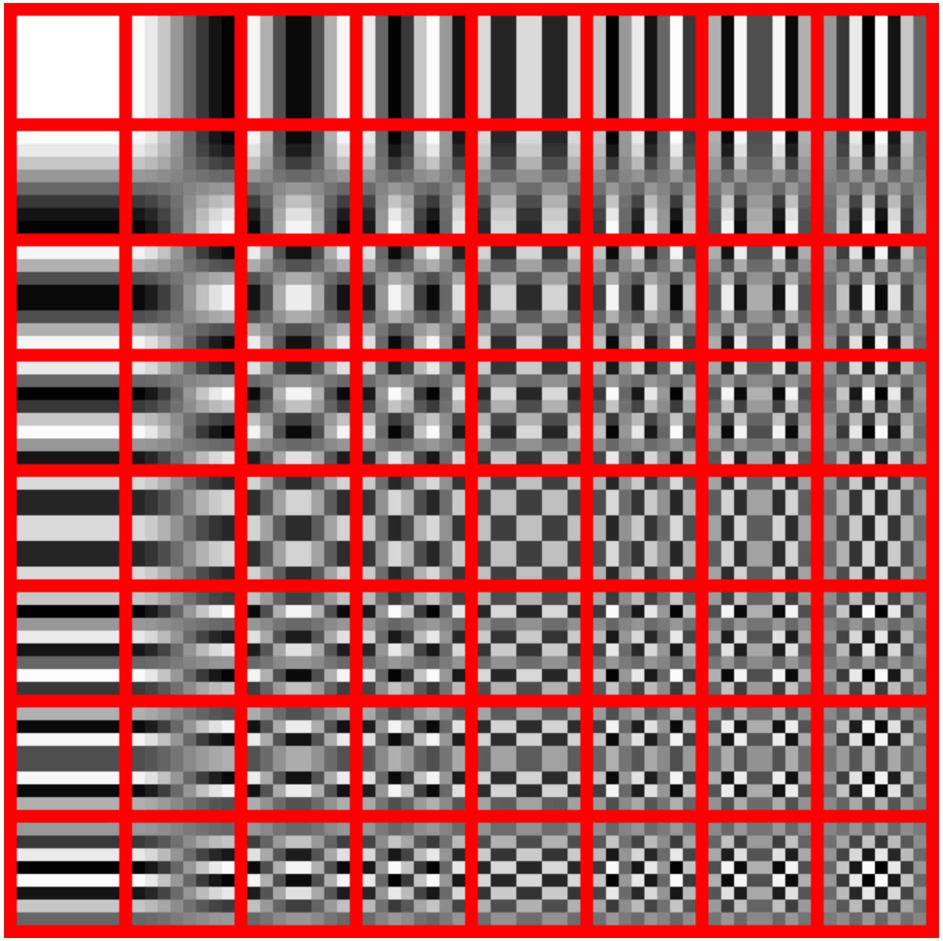
$$\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} = \boldsymbol{I}$$
$$\boldsymbol{\Phi}^{-1} = \boldsymbol{\Phi}^{\mathrm{T}}$$

- widely used
  - e.g. in JPEG as 8x8 DCT
- note:
  - there are other variants in use, e.g., where the orthogonality does not hold

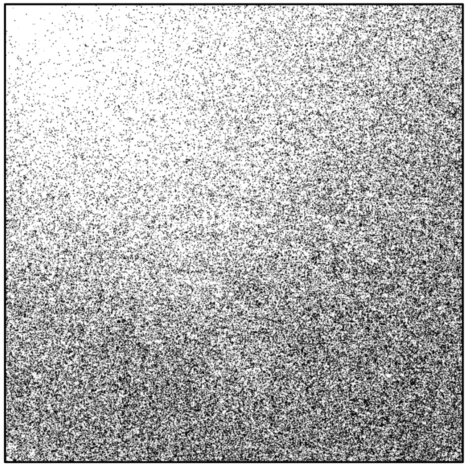
$$\Phi_{jk} = \cos\left(\frac{\pi}{N}\left(k + \frac{1}{2}\right)j\right)$$

  - fast algorithms available

DCT – Frequencies



concentrates energy in low order coefficients

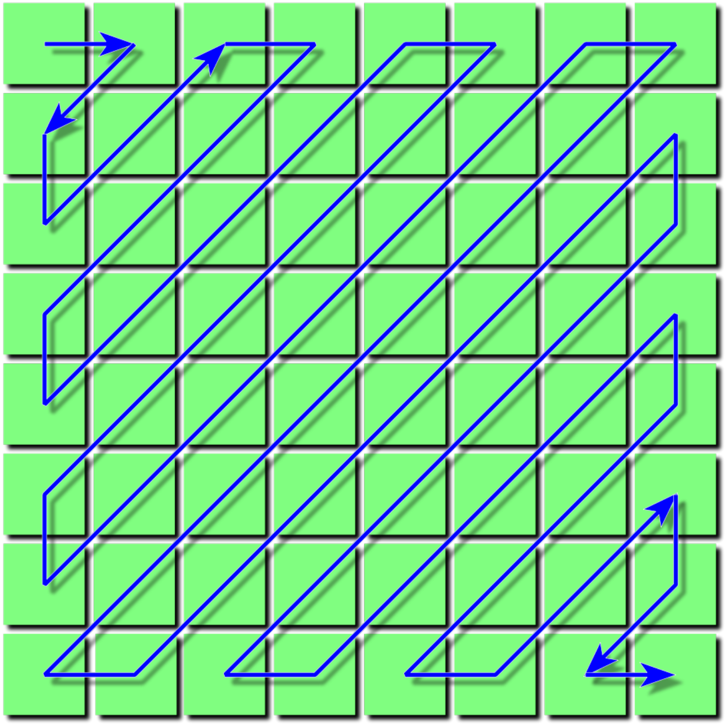


DCT, logarithmic scale – fully invertible

DCT – Selecting Coefficients

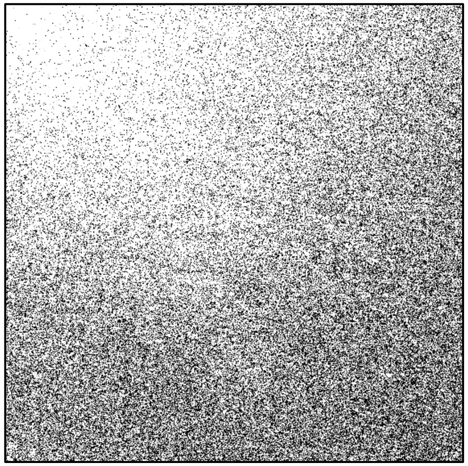
For 2D transformation: Select coefficients as features

Use frequencies in horizontal/vertical direction equally: Zig-Zag-Scan

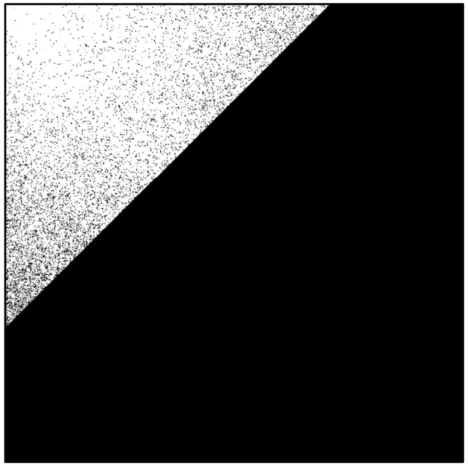


DCT – Examples

inverse DCT (full)

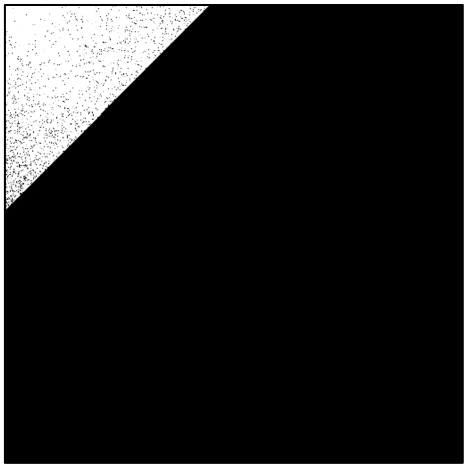


inverse DCT (using first 25% of DCT coefficients)

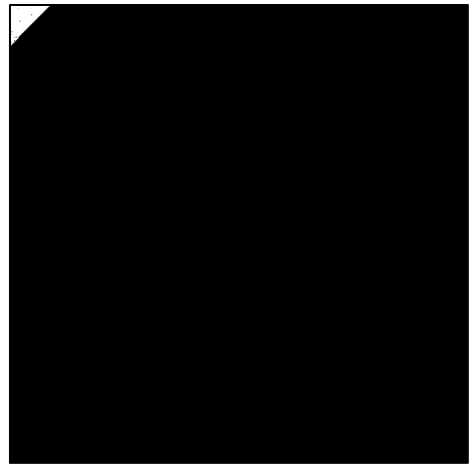


DCT – Examples

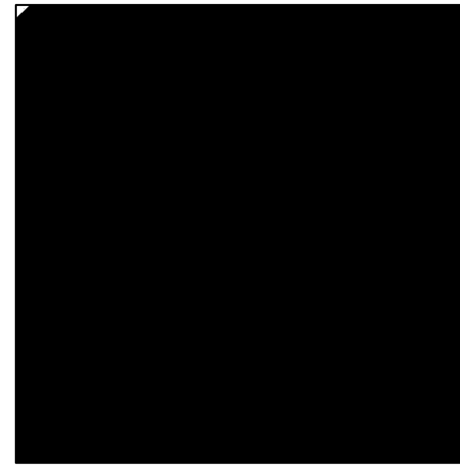
Inverse (10% of coefficients)



Inverse (1000 coefficients)



Inverse (100 coefficients)



References

slides based on

- slides of the lecture *Pattern Recognition* taught at the FAU Erlangen-Nuremberg, courtesy of D. Hahn, J. Hornegger, S. Steidl and E. Nöth.
- Ray Mooney: Support Vector Machines. Slides, University of Texas at Austin.
- Ch. Manning, P. Nayak: Introduction to Information Retrieval, Lecture 14: Support vector machines and machine learning documents. Stanford University.