Chapter 3 – Kernel Methods
Support Vector Machines

Maschinelles Lernen -
Grundverfahren WS20/21

Prof. Gerhard Neumann
KIT, Institut für Anthrophomatik und Robotik

A few announcements

- **For interested students: Meetup group for ML at KIT**
  https://www.meetup.com/de-DE/AI-Paper-Discussion-Group-ML-KA/
  - The Machine Learning Karlsruhe group (ML-KA) get together once per week and discuss a current research paper. The papers are mostly from the field of deep learning, computer vision, reinforcement lea…

- **Specific questions about the slides or the notebooks?**
  - Feel free to contact us
  - Also helps us to improve the content!
  - Any feedback is welcome

Learning Outcomes

What will we learn today?

- Understand the concept of Maximum Margin classifiers
- Define the corresponding optimization problem
- How do relax the problem using slack variables
- Connection to the hinge loss
- How do optimize the problem using sub-gradients

Agenda for Today

Recap: Linear Discriminators

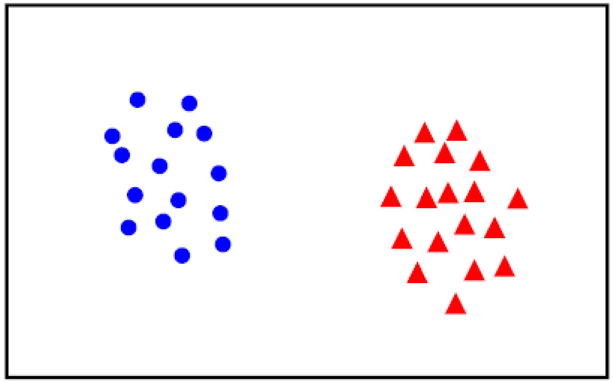
Support Vector Machines:

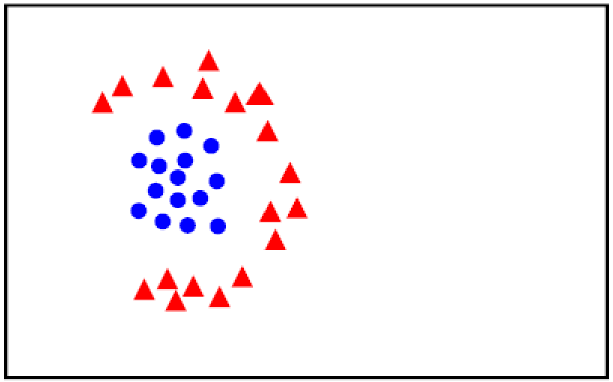
- Maximum Margin
- Optimization Problem
- Soft-Margin
- Hinge-Loss

Basics:

- Sub-gradients

Binary Classification: Previous Definition





Given the training data $(\boldsymbol{x}_i, y_i)$, i = 1…N, with $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{0, 1\}$, learn a classifier $f(\boldsymbol{x})$ such that:

$$f(\boldsymbol{x}_i) = \begin{cases} > 0, & \text{if } y_i = 1 \\ < 0, & \text{if } y_i = 0 \end{cases}$$

Binary Classification: New Definition

For SVMs, it simplifies notation to use +1 and -1 as class labels

New definition: Given the training data $(\boldsymbol{x}_i, y_i)$, i = 1…N, with $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$ learn a classifier $f(\boldsymbol{x})$ such that:

$$f(\boldsymbol{x}_i) = \begin{cases} > 0, & \text{if } y_i = 1 \\ < 0, & \text{if } y_i = -1 \end{cases}$$

Or: $f(\boldsymbol{x}_i)y_i > 0$ for a correct classification

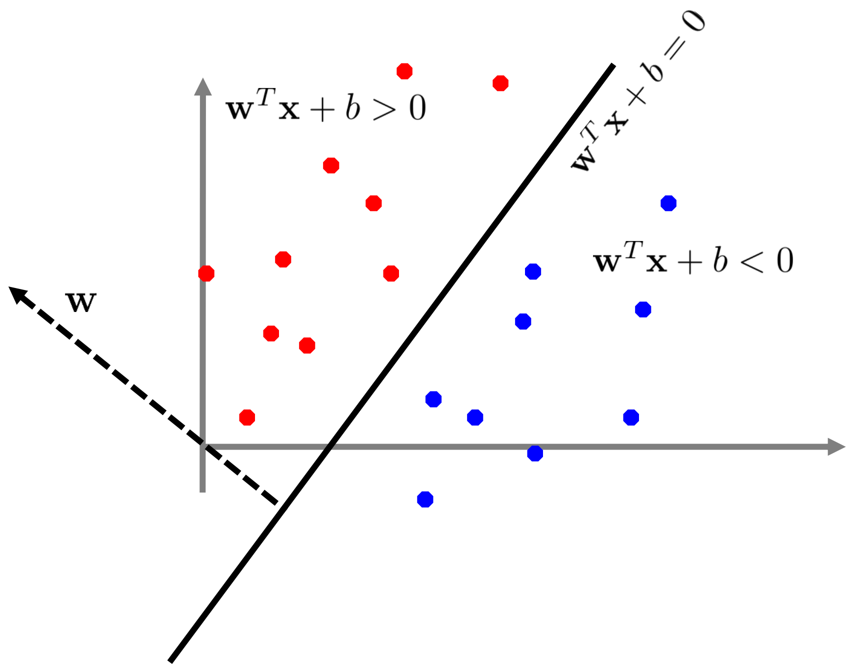
Recap: Linear Classifiers

A linear classifier is given in the form:

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

In 2D, the classifier is a line

- $\mathbf{w}$ is the normal to the line
- b is the bias



Basics: Projections of vectors

The scalar product of 2 vectors can be used to compute the projection of vector **a** on vector **b:**

- Geometric definition of scalar product

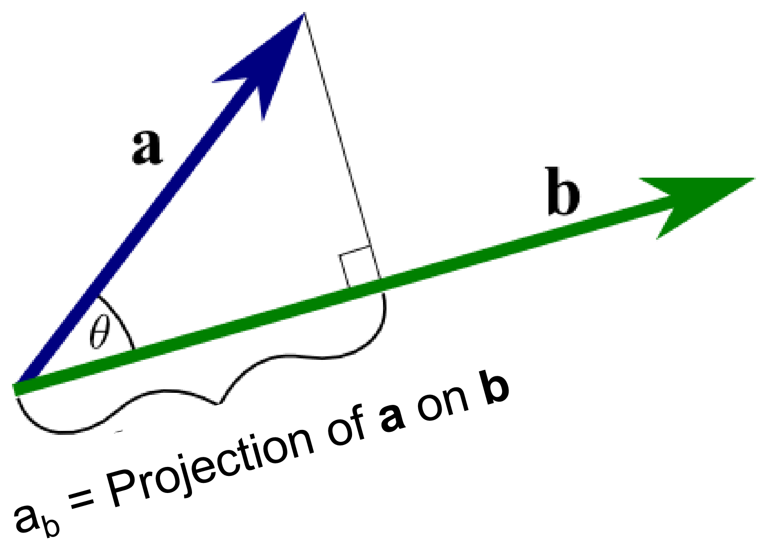
$$\boldsymbol{a}^T\boldsymbol{b} = \|\boldsymbol{a}\|\|\boldsymbol{b}\|\cos\theta$$

- Angle between 2 vectors

$$\cos\theta = \frac{\boldsymbol{a}^T\boldsymbol{b}}{\|\boldsymbol{a}\|\|\boldsymbol{b}\|}$$

- Scalar projection of **a** on **b**

$$a_b = \|\boldsymbol{a}\|\cos\theta = \frac{\boldsymbol{a}^T\boldsymbol{b}}{\|\boldsymbol{b}\|}$$

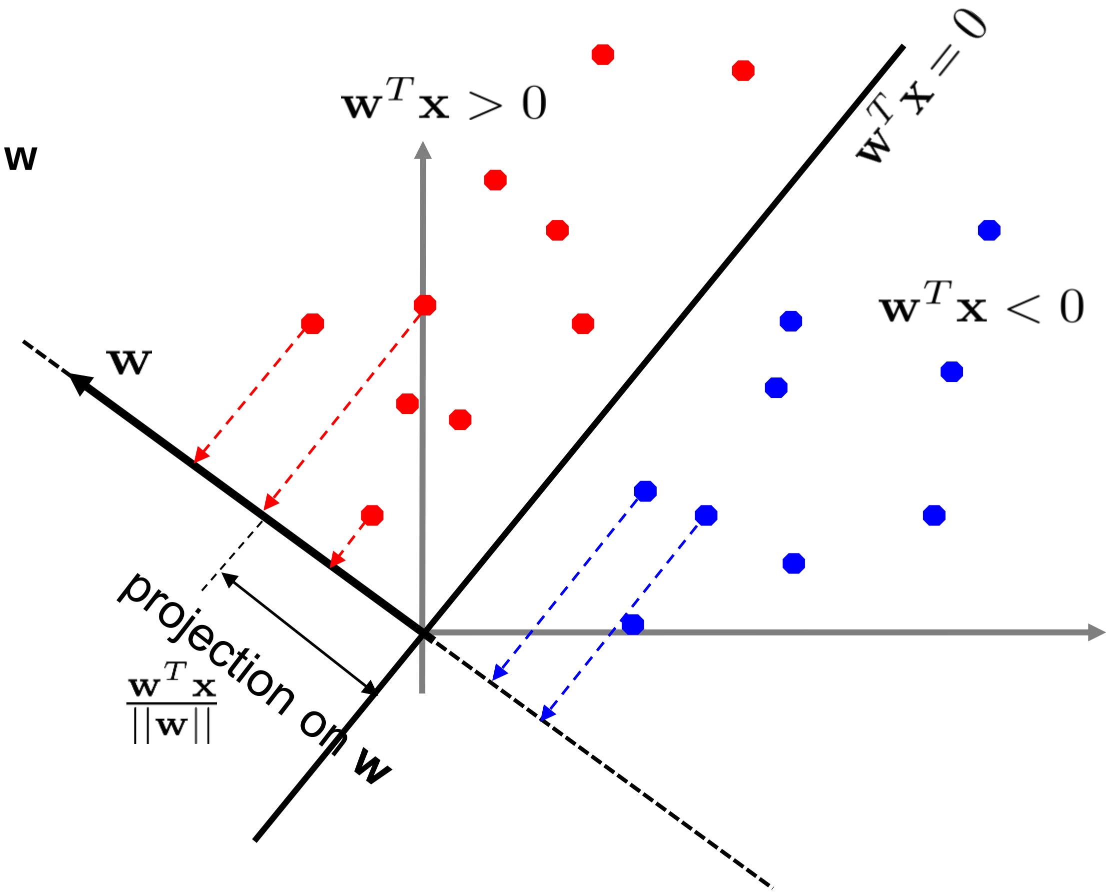


Observation: If 2 vectors are normal to each other, the projection is 0

Recap: Geometrical inspection

Observations:

- The decision boundary is normal to **w**
- Without b, it goes through the origin



Recap: Geometrical inspection

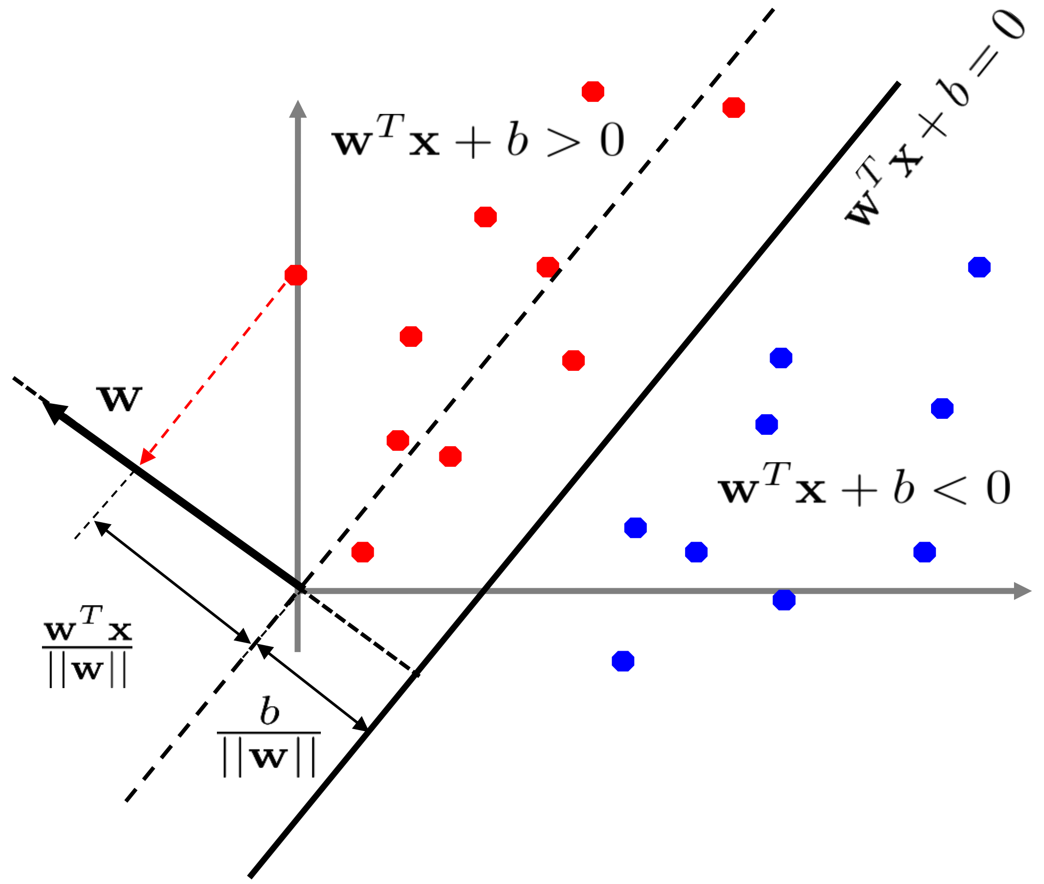
Observations:

- b shifts the decision boundary along (negative) direction of **w**
- The shift corresponds to adding $\frac{b}{||\mathbf{w}||}$ to the projection

$$\mathbf{w}^T\mathbf{x} + b = 0 \;\Rightarrow\; \underbrace{\frac{\mathbf{w}^T\mathbf{x}}{\|\mathbf{w}\|}}_{\text{projection } x_w} + \frac{b}{\|\mathbf{w}\|} = 0$$

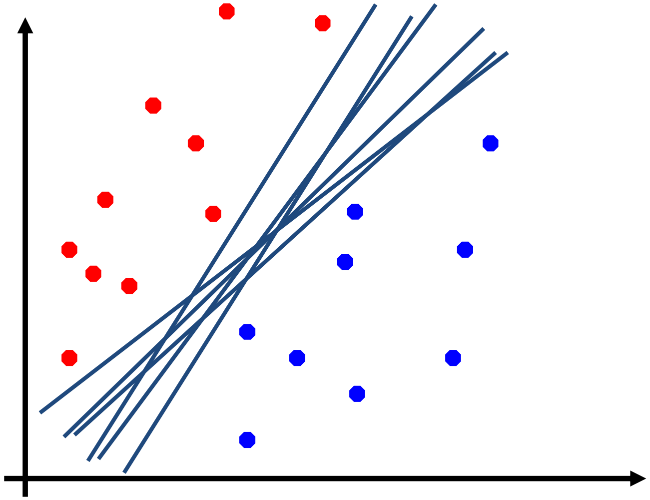
- I.e. in order for **x** to be on the decision boundary, the projection has to be

$$x_w = -\frac{b}{\|\mathbf{w}\|}$$



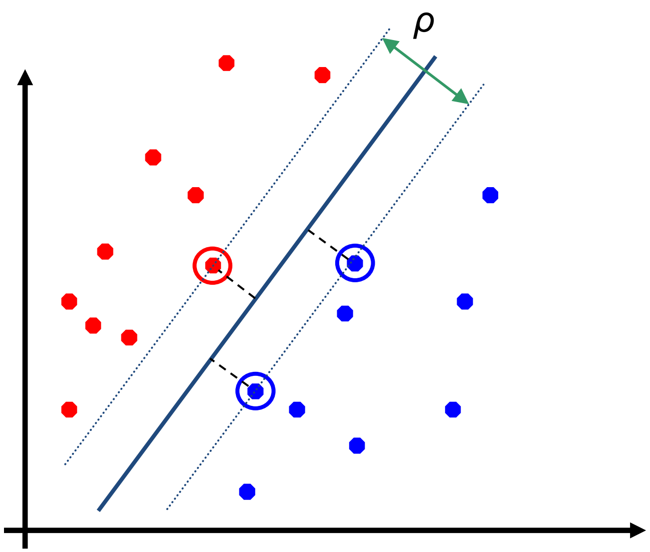
Optimal Separation

Which is the optimal line?



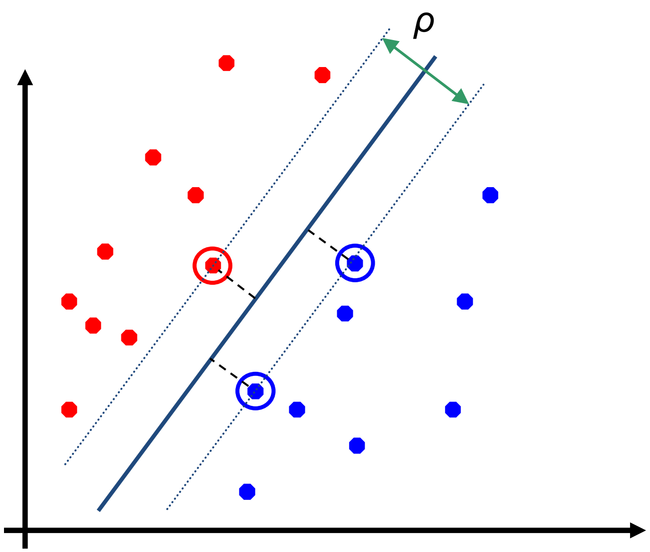
Maximum Margin

- **Support Vectors:** Data points closes to the decision boundary
  - Other examples can be ignored
- **Margin** ρ is the distance between the support vectors and the decision boundary
- Margin should be maximized
  - I.e. minimum distance between decision boundary and examples should be maximized



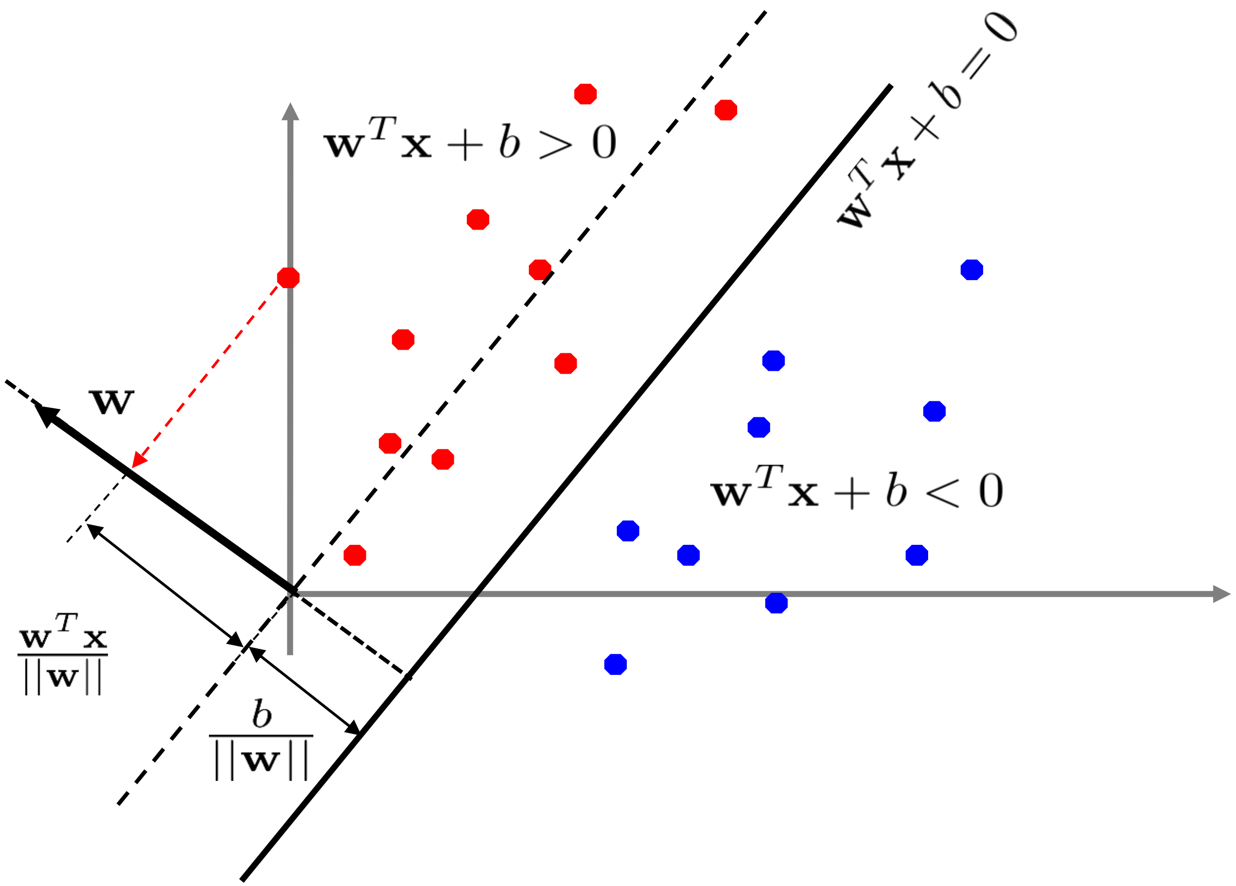
Maximum Margin

- **Maximize distance** between hyper-plane and “difficult examples”
  - Examples next to decision boundary
  - Also called Support Vectors
- **Intuition:**
  - Less examples close to decision boundary -> less uncertainty
- **Statistical Learning Theory:**
  - Maximum Margin Classifier has smaller complexity (VC-dimension)
  - And therefore generalizes better



Geometric Inspection

Distance between point $\mathbf{x}_i$ and line: $r = \dfrac{\mathbf{w}^T\mathbf{x}_i + b}{\|\mathbf{w}\|}$



Mathematical Formulation

Observation:

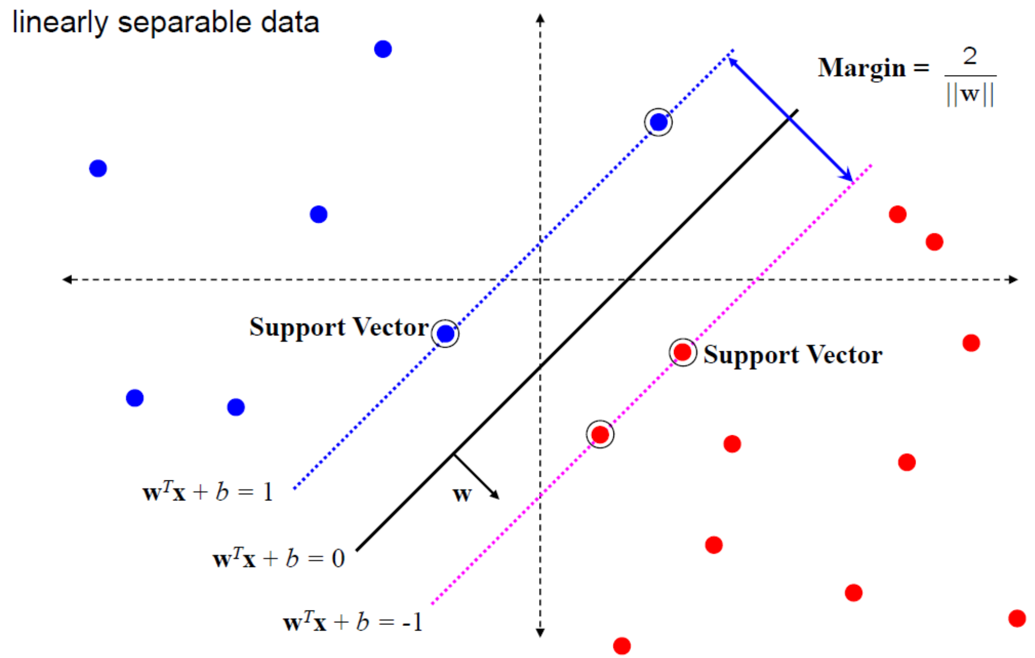
- $\mathbf{w}^T\mathbf{x} + b = 0$ and $c(\mathbf{w}^T\mathbf{x} + b) = 0$ define the same hyper-plane
- Scaling c can be chosen freely

Choose scaling such that

- For positive support vectors
  $\mathbf{w}^T\mathbf{x}_+ + b = +1$
- For negative support vectors
  $\mathbf{w}^T\mathbf{x}_- + b = -1$

Margin is then given by

$$\frac{\mathbf{w}^T\mathbf{x}_+ + b}{||\mathbf{w}||} - \frac{\mathbf{w}^T\mathbf{x}_- + b}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||}$$



SVM Optimization

Optimization problem:

$$\text{argmax}_{\mathbf{w}} \quad \frac{2}{||\mathbf{w}||},$$

Maximize margin

$$\text{s.t.} \quad \mathbf{w}^T\mathbf{x}_i + b \begin{cases} \geq +1, & \text{if } y_i = +1 \\ \leq -1, & \text{if } y_i = -1 \end{cases}$$

Condition for margin

Observations:

- If the constraints are not satisfied, our definition of the margin would be wrong, i.e,

$$\min_{\mathbf{x}_+ \in \boldsymbol{X}_+} \left(\mathbf{w}^T\mathbf{x}_+ + b\right) = +1$$

Positive support vectors

$$\max_{\mathbf{x}_- \in \boldsymbol{X}_-} \left(\mathbf{w}^T\mathbf{x}_- + b\right) = -1$$

Negative support vectors

$\boldsymbol{X}_-$: negative examples

$\boldsymbol{X}_+$: positive examples

  - Support vectors have the smallest distance to the decision boundary
- There is at least one positive and one negative data point that satisfy the support vector condition exactly (i.e. equality instead of inequality) from above
  - Why? Because of the argmax! Norm of weight vector could be reduced otherwise

SVM Optimization

Optimization problem:

$$\begin{aligned} &\operatorname{argmax}_{\mathbf{w}} \quad \frac{2}{||\mathbf{w}||}, \\ &\text{s.t.} \quad \mathbf{w}^T\mathbf{x}_i + b \begin{cases} \geq +1, & \text{falls } y_i = +1 \\ \leq -1, & \text{falls } y_i = -1 \end{cases} \end{aligned}$$

Maximize margin

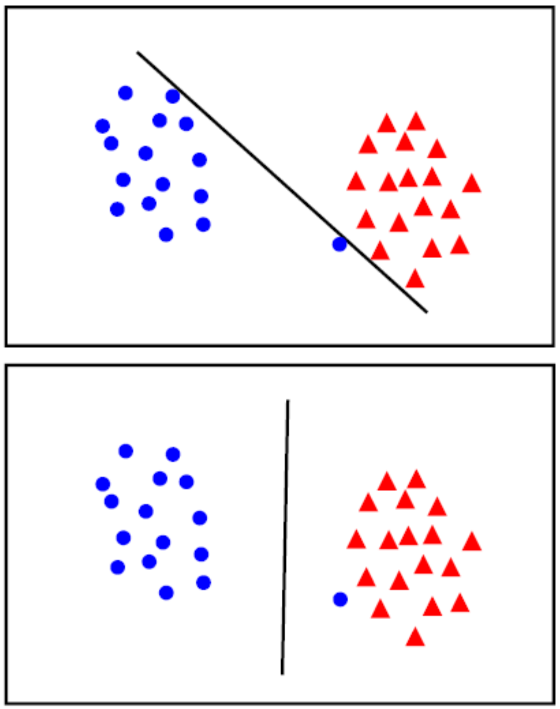
Condition for margin

Reformulation: Easier to solve, same solution

$$\left.\begin{aligned} &\operatorname{argmin}_{\mathbf{w}} \quad ||\mathbf{w}||^2, \\ &\text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \end{aligned}\right\}$$

- Quadratic optimization problem
- Linear Constraints
- Convex, single optimum

Back to linear Separability



What is the best w?

- Linear separable but: small margin

- Large margin but error in classification

We have to choose a trade-off between margin and classification accuracy!

Soft Max-Margin

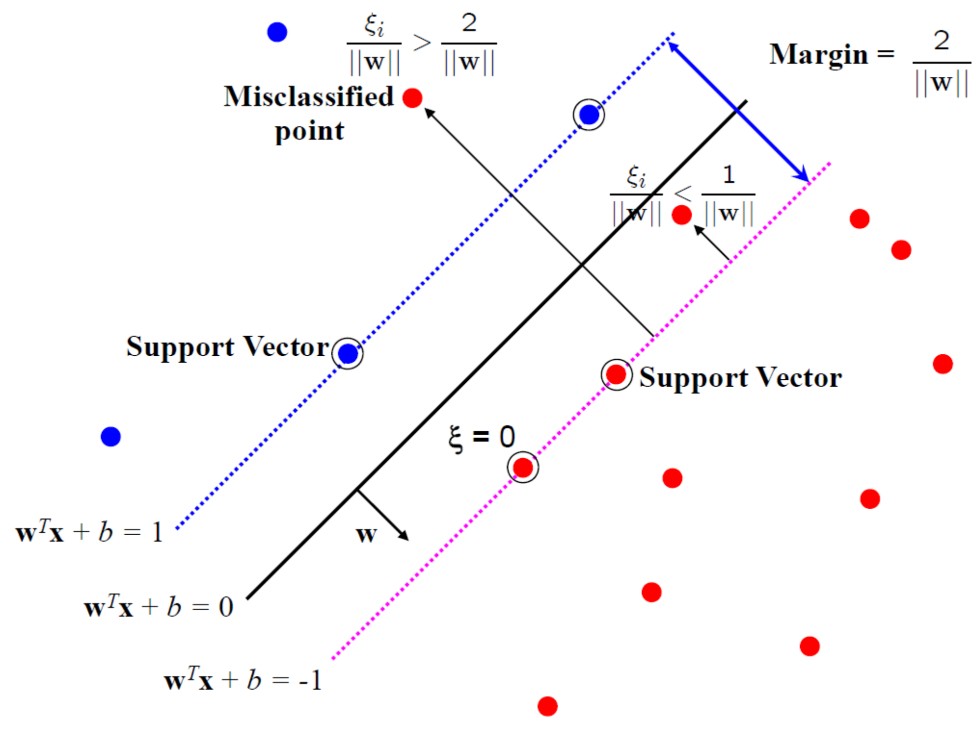
Introduce slack-variables:

$$\xi_i \geq 0$$

Allows violating the margin conditions

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$$

- $0 \leq \xi_i \leq 1$ sample is between margin and decision boundary: **margin violation**
- $\xi_i > 1$ sample is on the wrong side of the decision boundary: **misclassified**



Soft Max-Margin

Optimization problem:

$$\mathrm{argmin}_{\mathbf{w},\boldsymbol{\xi}} \quad ||\mathbf{w}||^2 + C\sum_i^N \xi_i,$$

Punish large slack variables

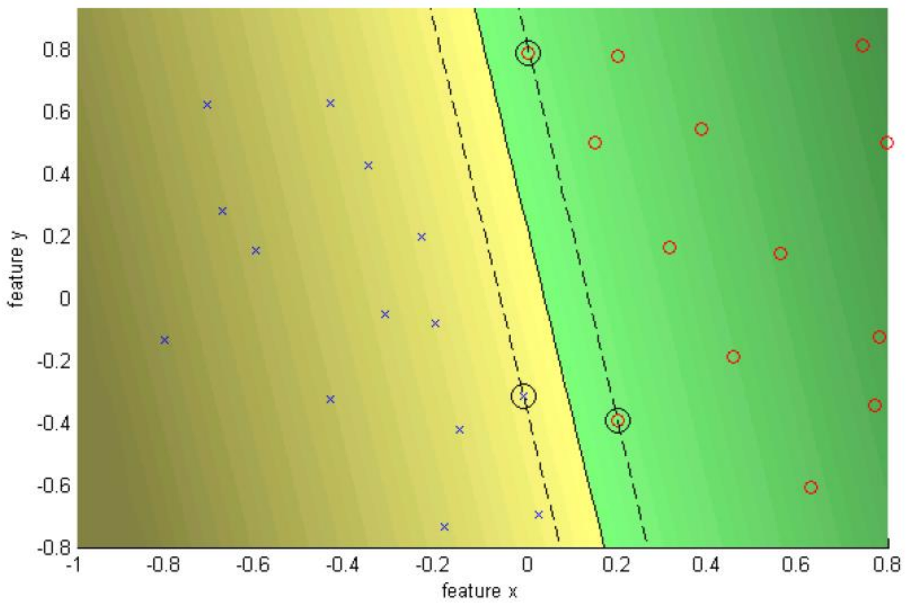
$$\text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

Condition for soft-margin

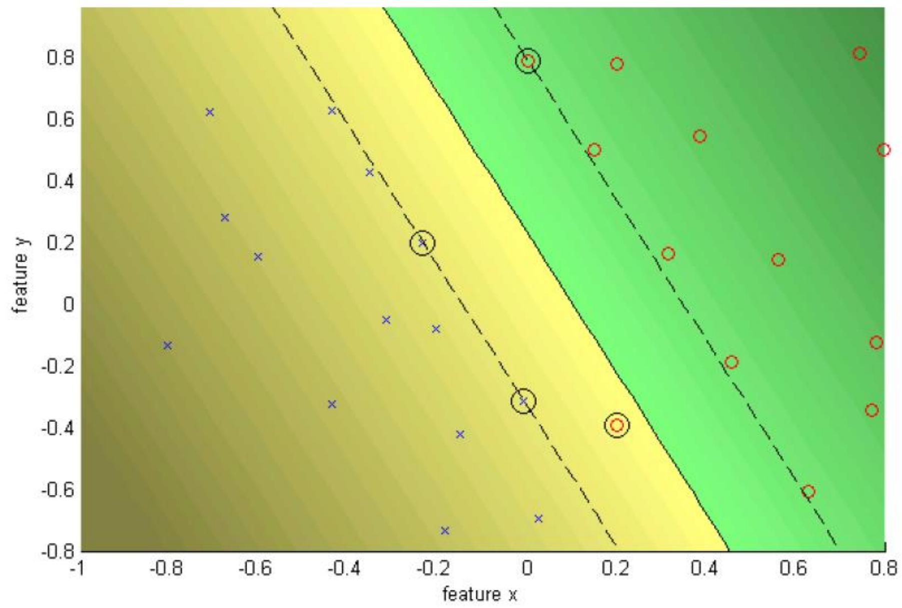
- C is a (inverse) regularization parameter
  - Small C: Constraints have little influence -> large margin -> large regularization
  - Large C: Constraints have large influence -> small margin -> small regularization
  - C infinite: Constraints are enforced-> hard margin -> no regularization

Illustration

Hard Margin



Soft Margin



Optimization

Constrained optimization:

$$\operatorname{argmin}_{\mathbf{w},\boldsymbol{\xi}} \quad ||\mathbf{w}||^2 + \underbrace{C}_{\text{Regularization parameter}} \sum_i^N \overbrace{\xi_i}^{\text{Slack variables}}, \quad \text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

SVMs can be reformulated into an unconstrained optimization problem

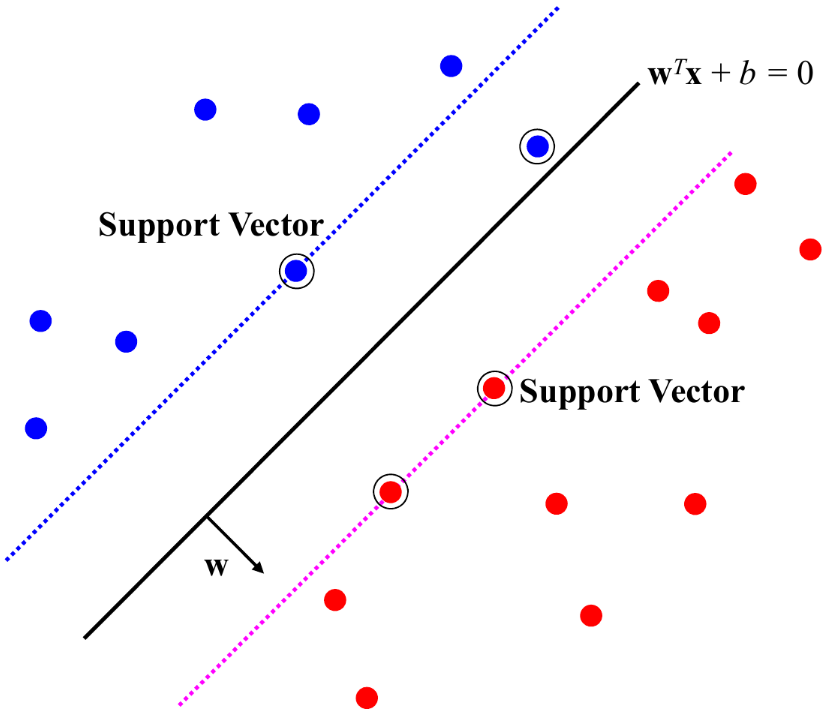
- Rewrite constraints: $\xi_i \geq \; 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1 - y_i f(\boldsymbol{x}_i)$
- Together with $\xi_i \geq 0$ this results in $\xi_i = \max\left(0, 1 - y_i f(\boldsymbol{x}_i)\right)$ (given that ξ_i should be minimized)

Unconstrained optimization (over w):

$$\operatorname{argmin}_{\mathbf{w}} \quad \underbrace{||\mathbf{w}||^2}_{\text{regularization}} \quad + \quad \underbrace{C \sum_{i=1}^N \max\left(0, 1 - y_i f(\boldsymbol{x}_i)\right)}_{\text{loss function}}$$

Hinge Loss

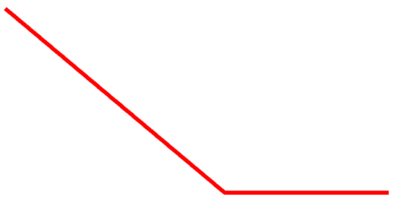
$$\operatorname{argmin}_{\mathbf{w}} \quad \underbrace{||\mathbf{w}||^2}_{\text{regularization}} + \underbrace{C \sum_{i=1}^{N} \max\left(0, 1 - y_i f(\boldsymbol{x}_i)\right)}_{\text{loss function}}$$



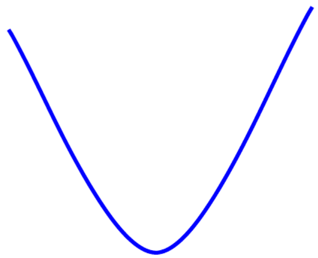
Points are in 3 categories:

- $y_i f(\boldsymbol{x}_i) > 1$: Point outside margin, no contribution to loss
- $y_i f(\boldsymbol{x}_i) = 1$: Point is on the margin, no contribution to loss as in hard margin
- $y_i f(\boldsymbol{x}_i) \leq 1$: Point violates the margin, contributes to loss

Loss function is convex



+



$$\operatorname{argmin}_{\mathbf{w}} \quad \underbrace{C \underbrace{\sum_{i=1}^{N} \max\left(0, 1 - y_i f(\boldsymbol{x}_i)\right)}_{\text{loss function}} + \underbrace{||\mathbf{w}||^2}_{\text{regularization}}}_{\text{Convex}}$$

- There is only one minimum
- We can find it with gradient descent
- **However:** Hinge loss is **not differentiable!**

Comparison to logistic loss function

- **SVM-hinge loss**

$$\text{argmin}_{\mathbf{w}} \quad \lambda \underbrace{||\mathbf{w}||^2}_{\text{regularization}} \quad + \underbrace{\sum_{i=1}^{N} \max\left(0, 1 - y_i f(\boldsymbol{x}_i)\right)}_{\text{data loss}}, \quad \text{with } \lambda = \frac{1}{C}$$

- **(Regularized) logistic regression loss (see lecture 2)**

$$\text{argmax}_{\mathbf{w}} \quad -\lambda||\mathbf{w}||^2 + \sum_{i=1}^{N} c_i \log(\sigma\big(f(\boldsymbol{x}_i)\big) + (1 - c_i) \log(1 - \sigma\big(f(\boldsymbol{x}_i)\big)), \quad \text{with } c_i \in \{0, 1\}$$

$$= \dots$$

$$= \text{argmin}_{\mathbf{w}} \quad \lambda \underbrace{||\mathbf{w}||^2}_{\text{regularization}} \quad + \underbrace{\sum_{i=1}^{N} \log\left(1 + \exp(-y_i f(\boldsymbol{x}_i))\right)}_{\text{data loss}}, \quad \text{with } y_i \in \{-1, 1\}$$

Both loss functions have similar interpretations

- Keep weights small + $y_i f(\boldsymbol{x}_i)$ should be large
- Saturates if $y_i f(\boldsymbol{x}_i)$ gets too large

Comparison to logistic loss function

SVM (hinge) loss:

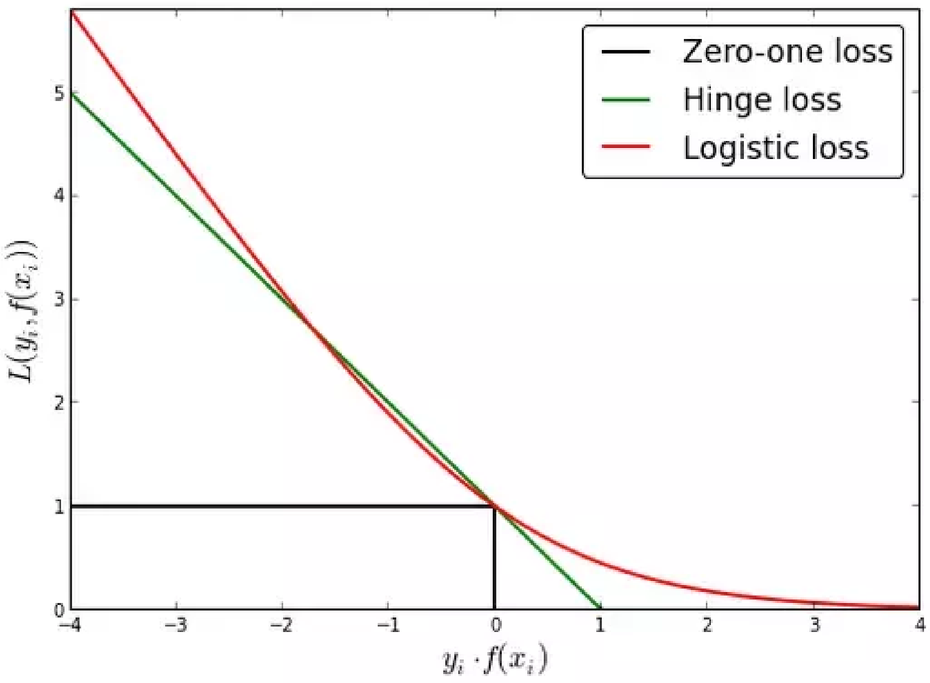
$$\max\left(0, 1 - y_i f(\boldsymbol{x}_i)\right)$$

- Outputs -1 or 1
- Estimates maximum margin solution
- Loss contribution is 0 for correct classification

Logistic loss:

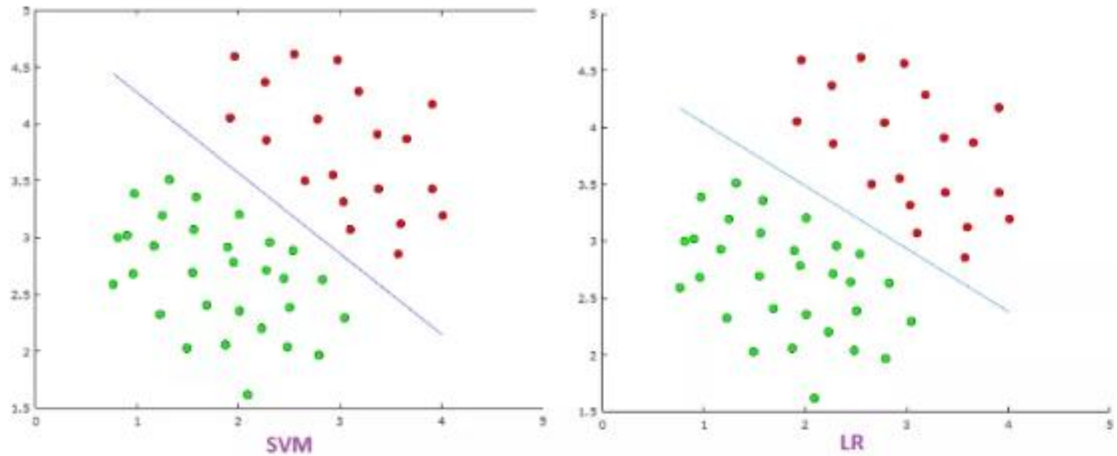
$$\log\left(1 + \exp(-y_i f(\boldsymbol{x}_i))\right)$$

- Outputs probabilities
- Contribution never 0
  - Often results in slightly less accurate classification
- Diverges faster than hinge loss
  - More sensitive to outliers

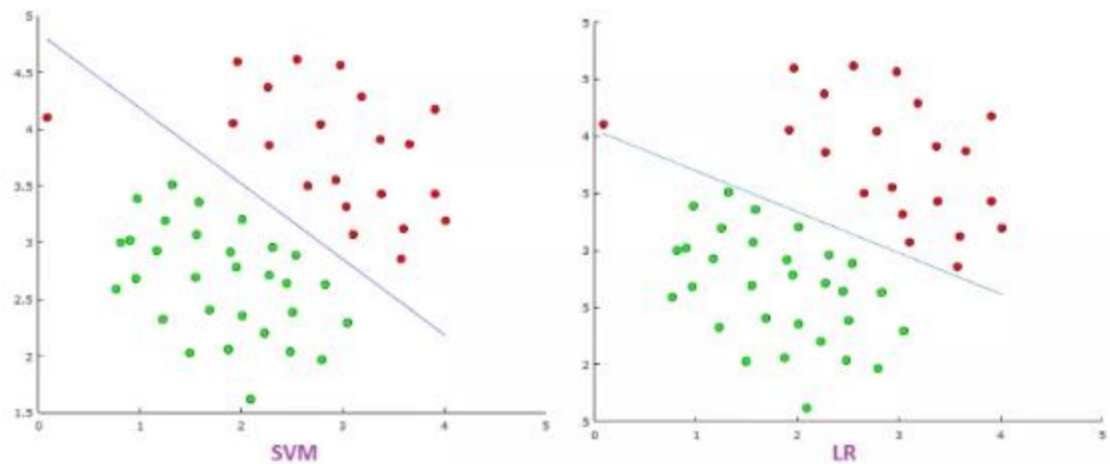


Comparison to logistic regression (LR)

- SVM finds more balanced decision boundary



- SVM is less sensitive to outliers



Agenda for Today

Recap: Linear Discriminators

Support Vector Machines:

- Maximum Margin
- Optimization Problem
- Soft-Margin
- Hinge-Loss

Basics:

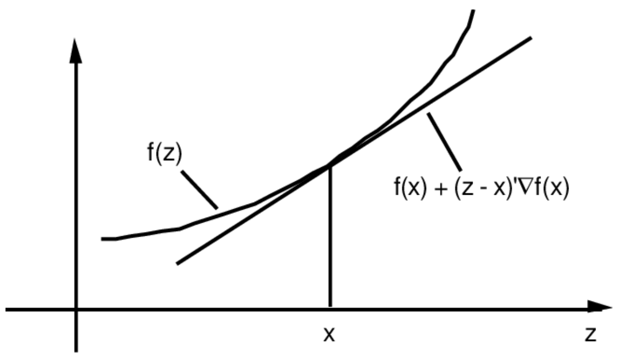
- Sub-gradients

Basics: Sub-gradients

Remember: For any convex function $f : \mathbb{R}^d \to \mathbb{R}$

$$f\left(\boldsymbol{z}\right) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T(\boldsymbol{z} - \boldsymbol{x})$$

- I.e. linear approximation underestimates function



A **subgradient** of a convex function *f* at point ***x*** is any ***g*** such that

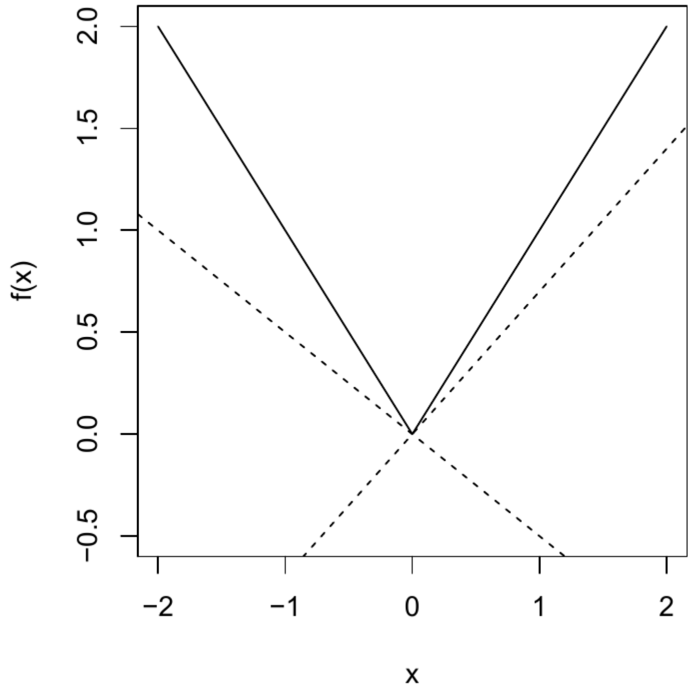
$$f\left(\boldsymbol{z}\right) \geq f(\boldsymbol{x}) + \boldsymbol{g}^T(\boldsymbol{z} - \boldsymbol{x})$$

- Always exists (also if *f* is not differentiable)
- If *f* is differentiable at ***x***, then $\boldsymbol{g} = \nabla f(\boldsymbol{x})$

Examples

Consider $f(x) = |x|$

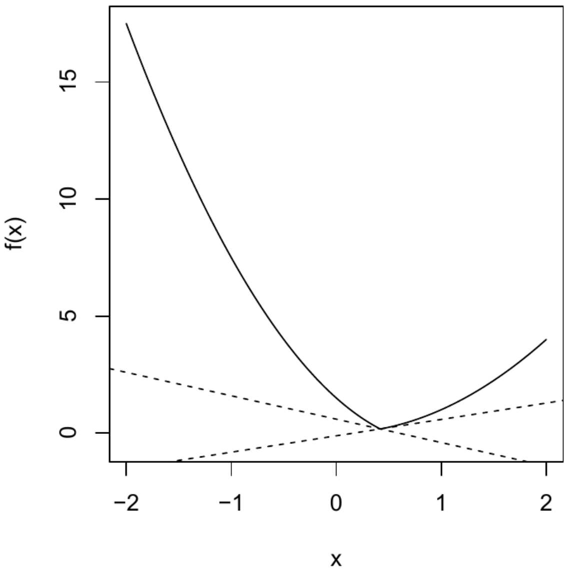
- For $x \neq 0$, unique sub-gradient of $g = \mathrm{sign}(x)$
- For $x = 0$, sub-gradient is any element of $[-1, 1]$



Examples

Let $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$ be convex, differentiable, and consider $f(x) = \max\{f_1(x), f_2(x)\}$

- For $f_1(x) > f_2(x)$, unique subgradient $g = \nabla f_1(x)$
- For $f_2(x) > f_1(x)$, unique subgradient $g = \nabla f_2(x)$
- For $f_1(x) = f_2(x)$, subgradient g is any point on the line segment between $\nabla f_1(x)$ and $\nabla f_2(x)$



Sub-Gradient Method

Like gradient descent, but replacing gradients with sub-gradients

Sub-gradient Descent:

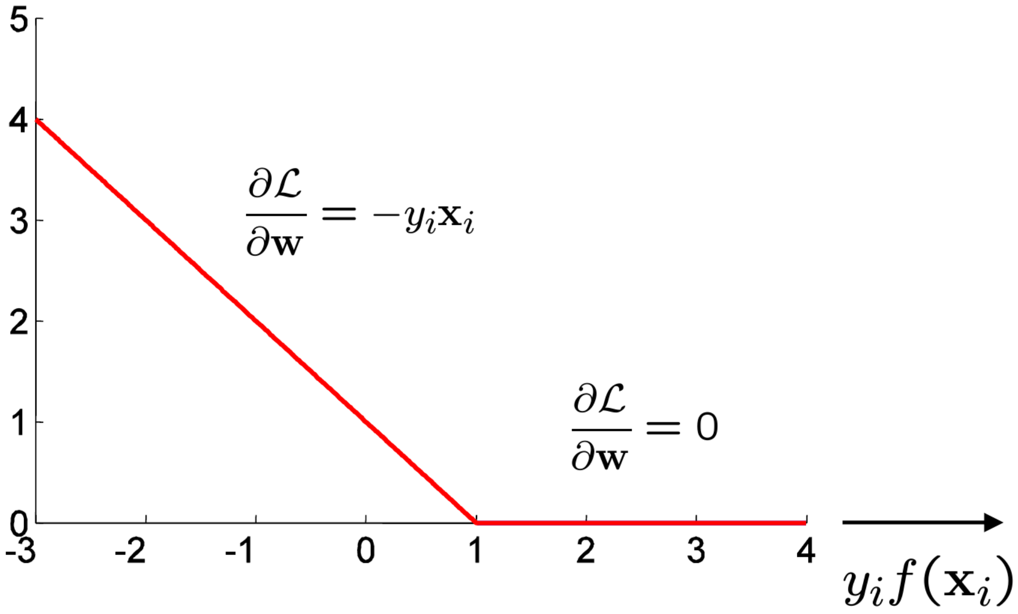
- Given convex *f*, not necessarily differentiable
- Initialize $\boldsymbol{x}_0$
- Repeat: $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \eta \boldsymbol{g}$, where $\boldsymbol{g}$ is any sub-gradient of f at point $\boldsymbol{x}_t$

Notes:

- Sub-gradients do not necessarily decrease *f* at every step (no real descent method)
- We need to keep track of the best iterate $\boldsymbol{x}^*$

Sub-gradients for hinge loss

$$\mathcal{L}(\mathbf{x}_i, y_i; \mathbf{w}) = \max(0, 1 - y_i f(\mathbf{x}_i)) \qquad f(\mathbf{x}_i) = \mathbf{w}^\top \mathbf{x}_i + b$$



Sub-gradient descent for SVMs

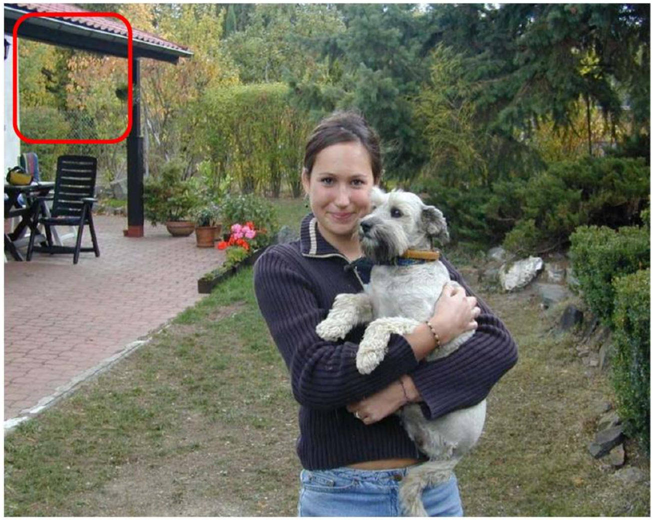
$$\text{argmin}_{\mathbf{w}} \quad \underbrace{C \sum_{i=1}^{N} \max\left(0, 1 - y_i f(\boldsymbol{x}_i)\right)}_{\text{loss function}} + \underbrace{||\mathbf{w}||^2}_{\text{regularization}}$$

At each iteration, pick random training sample $(\boldsymbol{x}_i, y_i)$

- If $y_i f(\boldsymbol{x}_i) < 1$: $\quad \boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta(2\boldsymbol{w}_t - C y_i \boldsymbol{x}_i)$
- Otherwise: $\quad \boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta 2\boldsymbol{w}_t$

Application: Pedestrian Tracking

Objective: Detect (localize) standing humans in images



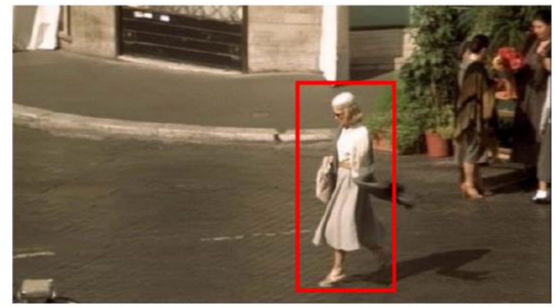
Detection with a sliding window approach:

- Reduces object detection to binary classification
- Does an image window contain a person or not?

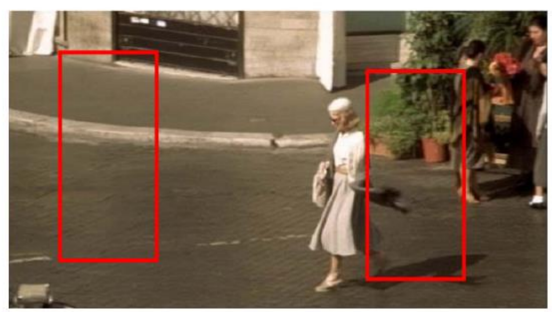
Training Data

- Positive Data: 1208 examples

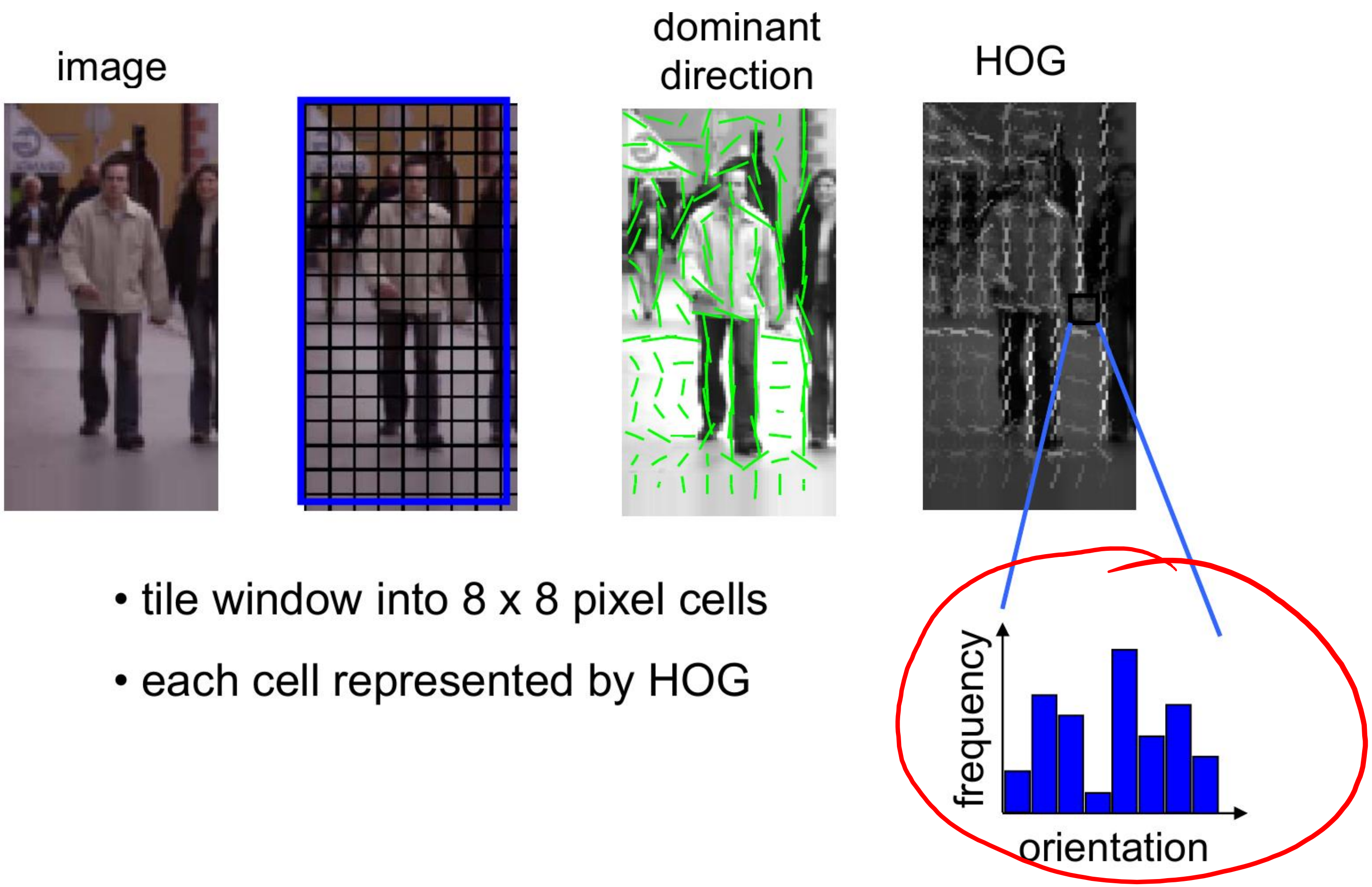


- Negative Data: 1218 examples

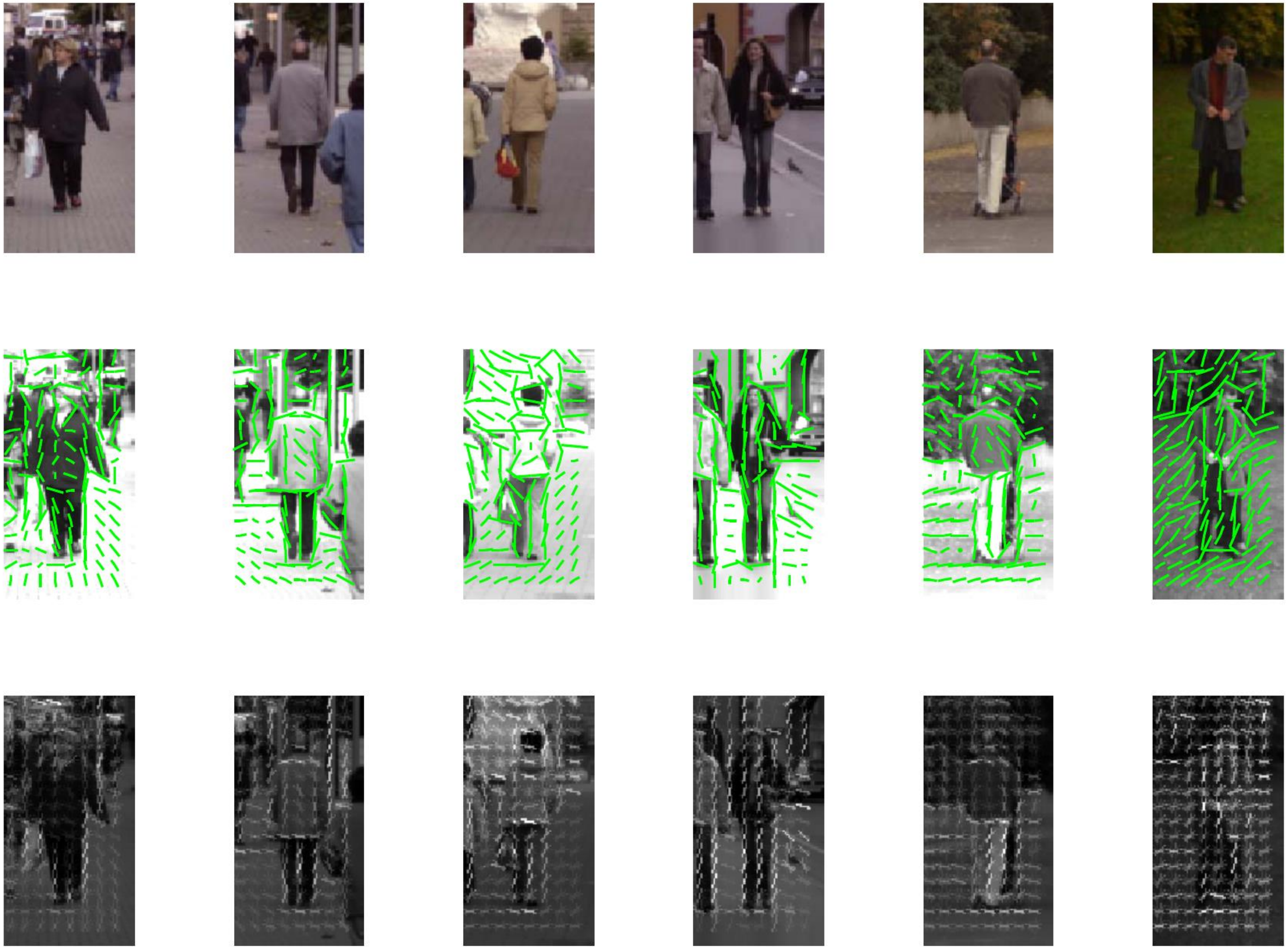
Features: Histogram of oriented Gradients (HoG features)



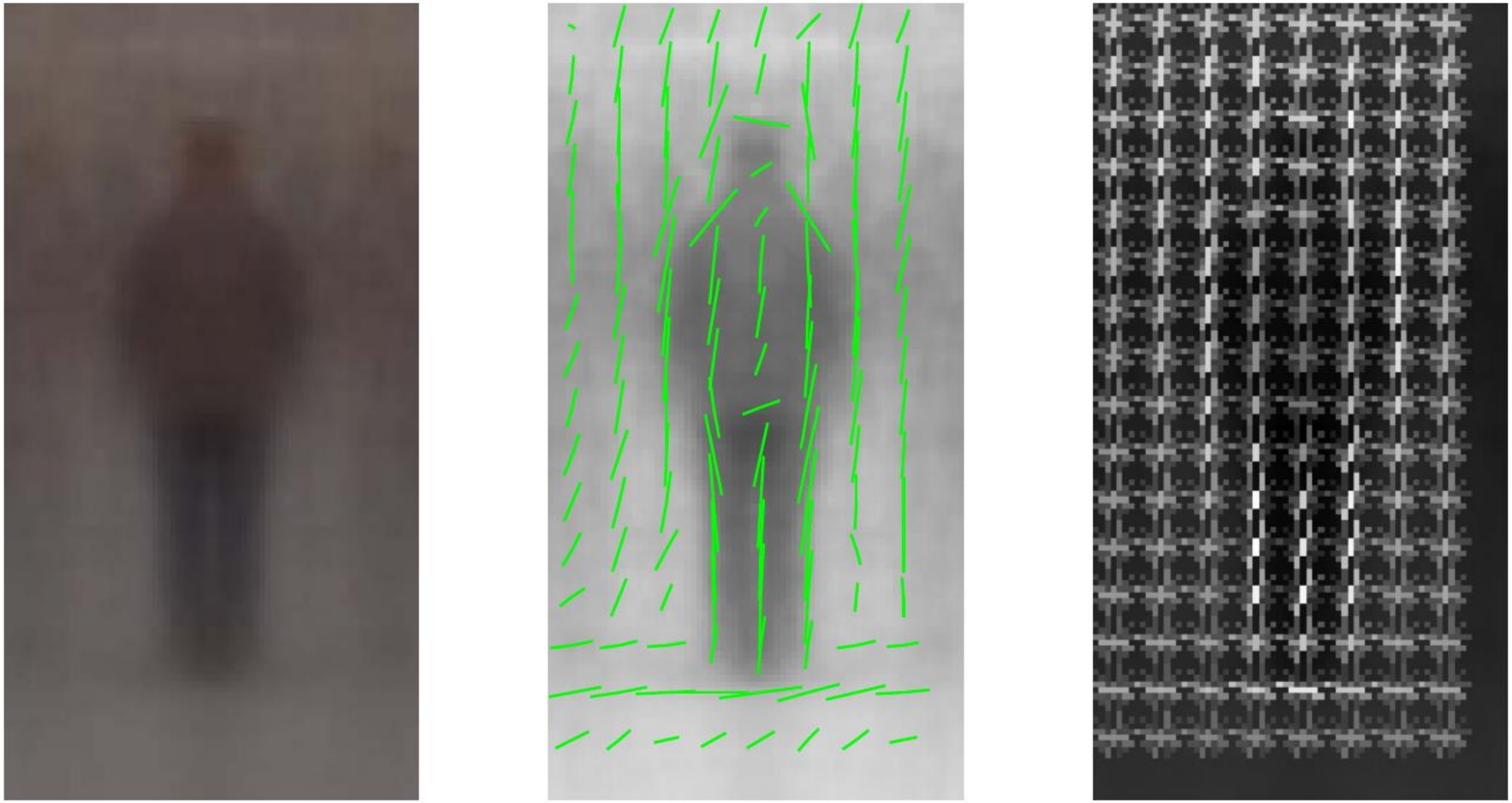
- tile window into 8 x 8 pixel cells
- each cell represented by HOG

Feature vector dimension = 16 x 8 (for tiling) x 8 (orientations) = 1024

Example HoG features



Averaged Positive Example

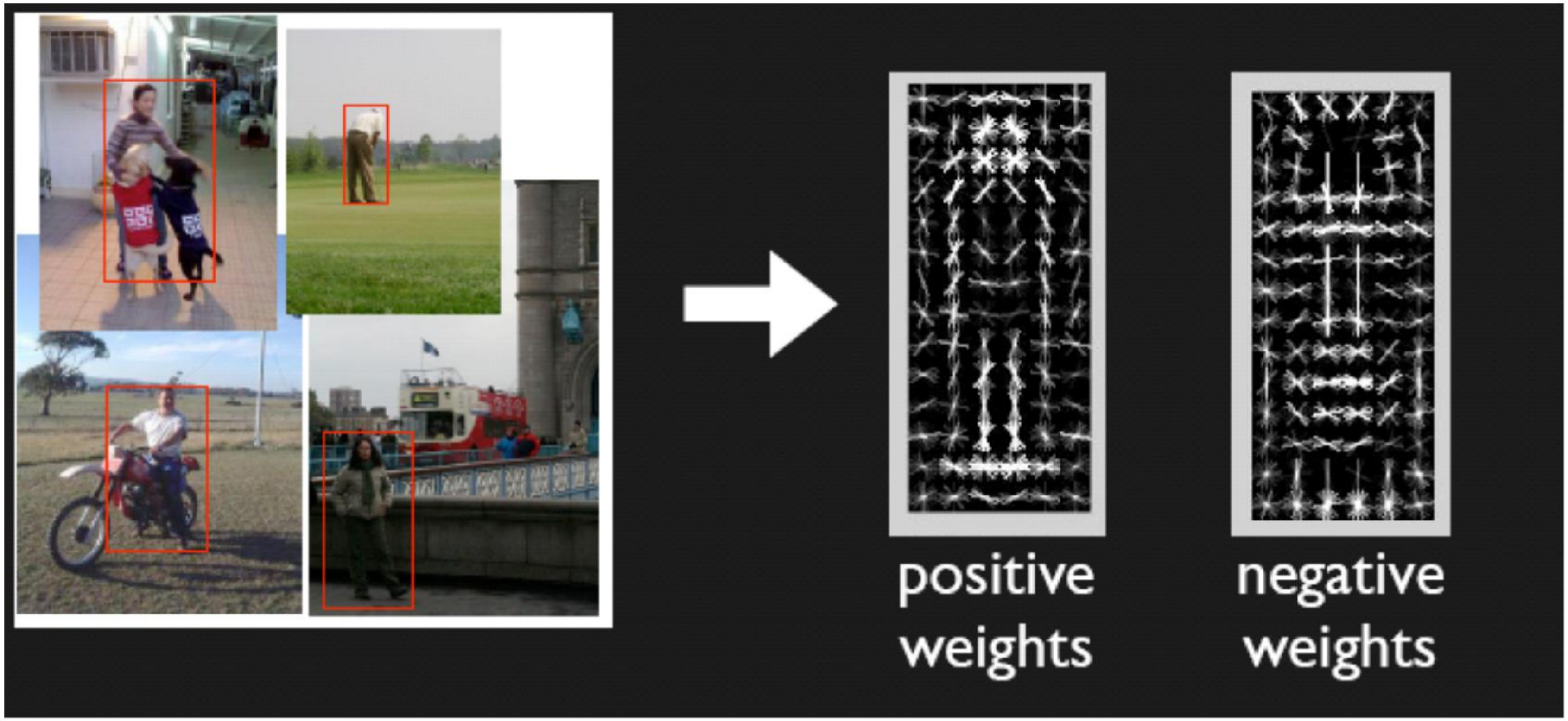


Example detection

Dalal and Triggs, CVPR 2005

Learned model

Model: $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$



Wrap-up

SVMs have been the "gold standard" in the 90s and 2000s for classification

- **SVM have been used successfully in many real-world problems**
  - text (and hypertext) categorization
  - image classification
  - bioinformatics (Protein classification, cancer classification)
  - hand-written character recognition

- **Can be extended to complex feature spaces using kernels (next part)**

- **… and regression problems (support vector regression, not covered)**

In the last 7-10 years, neural networks have outperformed SVMs on most applications

- However, similar insights are still used (e.g, hinge loss)

SVMs with Kernels

Support Vector Machines (continued…)

SVMs with features:

- Maximum margin principle
- Slack variables allow for margin violation

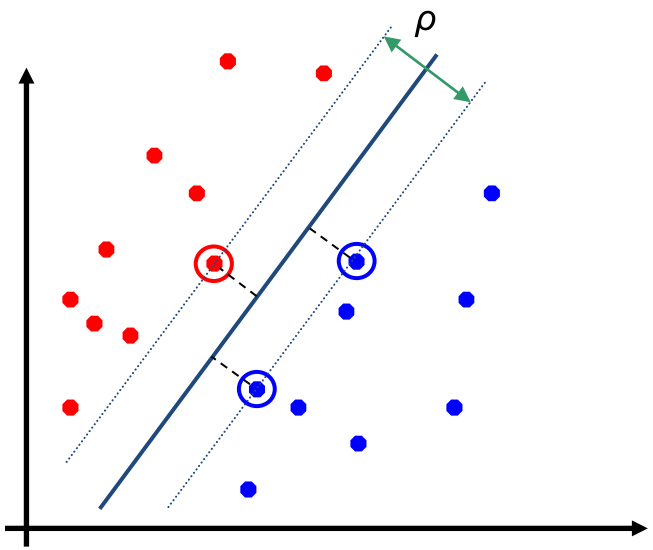
$$\operatorname{argmin}_{\mathbf{w},\boldsymbol{\xi}} \quad ||\mathbf{w}||^2 + C\sum_i^N \xi_i,$$

$$\text{s.t.} \quad y_i(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

- Simpler formulation without slack variables

$$\operatorname{argmin}_{\mathbf{w}} \quad ||\mathbf{w}||^2,$$

$$\text{s.t.} \quad y_i(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i) + b) \geq 1$$

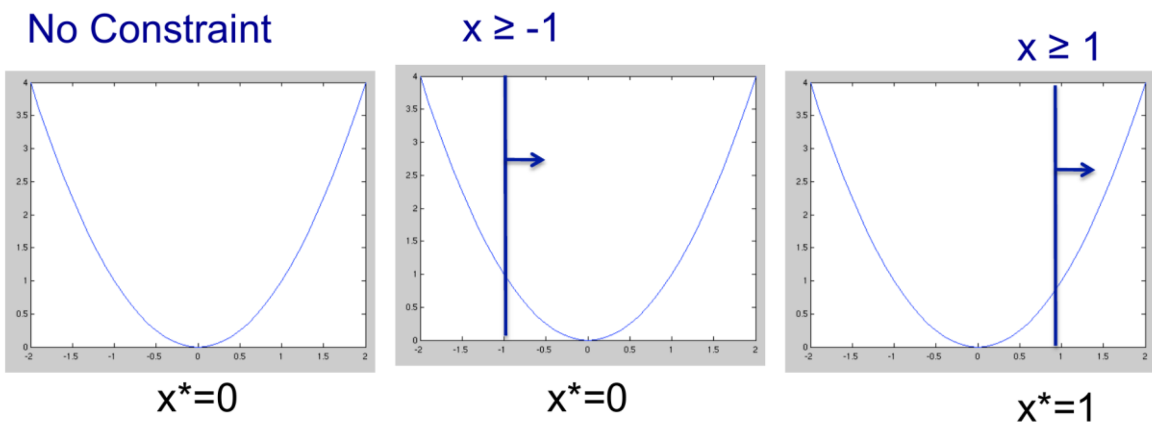


In order to apply the kernel trick, we need to apply our Constrained Optimization knowledge!

Recap: Constrained Optimization

Simple constrained optimization problem:

$$\arg\min_x x^2 \quad \text{s.t.} \quad x \geq b$$



How do we solve the constrained optimization problem? **Lagrangian Multipliers!**

Recap: General formulation

General Formulation:
$$\min_{\boldsymbol{x}} f(\boldsymbol{x}),$$
$$\text{s.t.} \quad h_i(\boldsymbol{x}) \geq b_i, \ \text{for } i = 1 \dots K$$

- Several inequality constraints (equality constraints also possible)

Lagrangian optimization:
$$\min_{\boldsymbol{x}} \max_{\boldsymbol{\lambda}} L(\boldsymbol{x}, \boldsymbol{\lambda}), \qquad L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \sum_{i=1}^{K} \lambda_i \big(h_i(\boldsymbol{x}) - b_i\big)$$
$$\text{s.t. } \lambda_i \geq 0, \ \text{for } i = 1 \dots K$$

Recap: Dual optimization

Primal optimization problem:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}),$$
$$\text{s.t.} \quad h_i(\boldsymbol{x}) \geq b_i, \;\; \text{for } i = 1 \dots K$$

Dual optimization problem:

$$\boldsymbol{\lambda}^* = \arg\max_{\boldsymbol{\lambda}} g(\boldsymbol{\lambda}), \quad g(\boldsymbol{\lambda}) = \min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda})$$
$$\text{s.t.} \quad \lambda_i \geq 0, \;\; \text{for } i = 1 \dots K$$

- g is also called the dual function of the optimization problem
- We essentially swapped min and max in the definition of L

Slaters condition: For a convex objective and convex constraints, solving the dual is equivalent to solving the primal!

- Optimal primal parameters can be obtained from optimal dual parameters, i.e.

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}^*)$$

Lagrangian Optimization

Basic "Cookbook":

1. Write down Lagrangian

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \sum_{i=1}^{K} \lambda_i \big(h_i(\boldsymbol{x}) - b_i\big)$$

2. Obtain optimal solution for primal parameters
   – Compute derivative, set to zero and solve for x

$$\frac{\partial L(\boldsymbol{x}, \boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}} = 0 \rightarrow \boldsymbol{x}^* = f(\boldsymbol{\lambda})$$

3. Set $\boldsymbol{x}^*$ back into Lagrangian to obtain the dual function

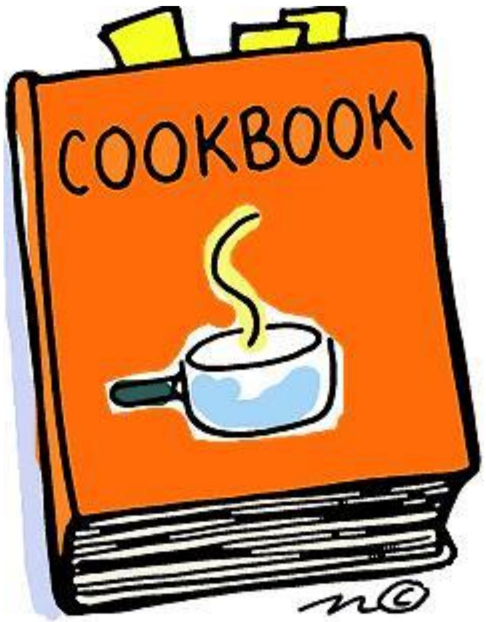
$$g(\boldsymbol{\lambda}) = L(f(\boldsymbol{\lambda}), \boldsymbol{\lambda})$$

4. Obtain optimal solution for the dual function
   – Set derivative to zero or gradient descent

$$\boldsymbol{\lambda}^* = \mathrm{argmax}_{\boldsymbol{\lambda}} g(\boldsymbol{\lambda}), \quad \text{s.t. } \lambda_i \geq 0, \forall i$$

1. Compute optimal primal parameters for given

$$\boldsymbol{x}^* = f(\boldsymbol{\lambda}^*)$$



Dual derivation of the SVM

SVM optimization:

- Lagrangian: $\text{argmin}_{\mathbf{w}} \quad ||\mathbf{w}||^2, \quad \text{s.t.} \quad y_i(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) + b) \geq 1$

Compute optimal *w*: $L(\boldsymbol{w}, \boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_i \lambda_i \big(y_i(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i) + b) - 1\big)$

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_i \lambda_i y_i \boldsymbol{\phi}(\boldsymbol{x}_i) = 0,$$

$$\boldsymbol{w}^* = \sum_i \lambda_i y_i \boldsymbol{\phi}(\boldsymbol{x}_i)$$

  - Many of the α_i will be zero (constraint satisfied)
  - If α_i is not zero, $\boldsymbol{\phi}(\boldsymbol{x}_i)$ is a support vector
  - The optimal weight vector w is a linear combination of the support vectors!

Dual derivation of the SVM

SVM optimization:

- Lagrangian: $\mathrm{argmin}_{\mathbf{w}} \quad ||\mathbf{w}||^2, \quad \text{s.t.} \quad y_i(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) + b) \geq 1$

Optimality condition for *b:*

$$L(\boldsymbol{w}, \boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_i \lambda_i \big(y_i(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i) + b) - 1\big)$$

- We do not obtain a solution for b
- But an additional condition for the lambdas

$$\frac{\partial L}{\partial b} = -\sum_i \lambda_i y_i \Rightarrow \sum_i \lambda_i y_i = 0$$

b can be computed from w:

- If $\lambda_i > 0$, then $\boldsymbol{x}_i$ is on the margin, i.e.:

$$y_i(\mathbf{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i) + b) = 1$$
$$y_i y_i(\mathbf{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i) + b) = y_i$$
$$b = y_i - \mathbf{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)$$

Kernel Trick in SVMs

Lagrangian: $$L(\boldsymbol{w}, \boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_i \lambda_i \Big(y_i(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i) + b) - 1\Big), \quad \boldsymbol{w}^* = \sum_i \lambda_i y_i \boldsymbol{\phi}(\boldsymbol{x}_i)$$

Dualfunction: $$\begin{aligned} g(\boldsymbol{\lambda}) &= L(\boldsymbol{w}^*, \boldsymbol{\lambda}) \\ &= \underbrace{\frac{1}{2}\sum_i\sum_j \lambda_i\lambda_j y_i y_j \boldsymbol{\phi}(\boldsymbol{x}_i)^T\boldsymbol{\phi}(\boldsymbol{x}_j)}_{\boldsymbol{w}^{*T}\boldsymbol{w}^*} - \sum_i \lambda_i y_i \Big(\underbrace{\sum_j \lambda_j y_j \boldsymbol{\phi}(\boldsymbol{x}_j)}_{\boldsymbol{w}^*}\Big)^T \boldsymbol{\phi}(\boldsymbol{x}_i) + \sum_i \lambda_i \\ &= \sum_i \lambda_i - \frac{1}{2}\sum_i\sum_j \lambda_i\lambda_j y_i y_j \boldsymbol{\phi}(\boldsymbol{x}_i)^T\boldsymbol{\phi}(\boldsymbol{x}_j) \end{aligned}$$

We just derived the kernel trick for SVMs

$$g(\boldsymbol{\lambda}) = \sum_i \lambda_i - \frac{1}{2}\sum_i\sum_j \lambda_i\lambda_j y_i y_j \boldsymbol{k}(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

- Scalar products of the feature vectors can be written as kernels

Kernelized SVM

Solve dual optimization problem:

$$\max_{\boldsymbol{\lambda}} \sum_i \lambda_i - \frac{1}{2} \sum_i \sum_j \lambda_i \lambda_j y_i y_j \boldsymbol{k}(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

$$\text{s.t. } \lambda_i \geq 0, \forall i \in [1 \dots N], \quad \sum_i \lambda_i y_i = 0$$

Compute primal from dual parameters

- Weight vector (can not be represented): $\boldsymbol{w}^* = \sum_i \lambda_i y_i \boldsymbol{\phi}(\boldsymbol{x}_i)$
- Bias: for any i with $\lambda_i > 0$: $$b = y_k - \boldsymbol{\phi}(\boldsymbol{x}_k)^T \boldsymbol{w}^* = y_k - \sum_i y_i \lambda_i k(\boldsymbol{x}_i, \boldsymbol{x}_k)$$
- Decision function: $$f(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{w}^* + b = \sum_i y_i \lambda_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b$$

Relaxed constraints with slack

Primal optimization problem:

$$\text{argmin}_{\mathbf{w},\boldsymbol{\xi}} \quad ||\mathbf{w}||^2 + C\sum_i^N \xi_i,$$

$$\text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

Dual optimization problem:

$$\max_{\boldsymbol{\lambda}} \sum_i \lambda_i - \frac{1}{2}\sum_i\sum_j \lambda_i\lambda_j y_i y_j \boldsymbol{k}(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

$$\text{s.t.} \;\; C \geq \lambda_i \geq 0, \forall i \in [1 \ldots N], \quad \sum_i \lambda_i y_i = 0$$

- For computing b, we now take an example where $C > \lambda_i > 0$

What changed?

- Added upper bound of C on λ_i !

Intuitive explanation:

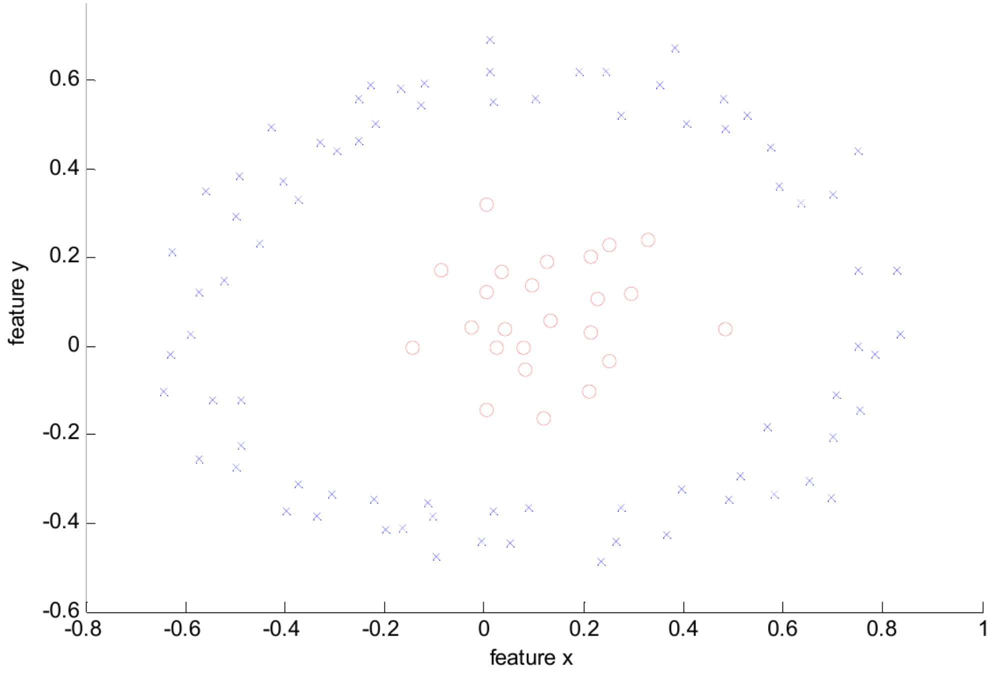
- Without slack, $\lambda_i \to \infty$ when constraints are violated (points misclassified)
- Upper bound of C limits the λ_i , so misclassifications are allowed

Example: SVM with RBF kernel

Data is non-linearly separable in feature space

RBF-kernel:

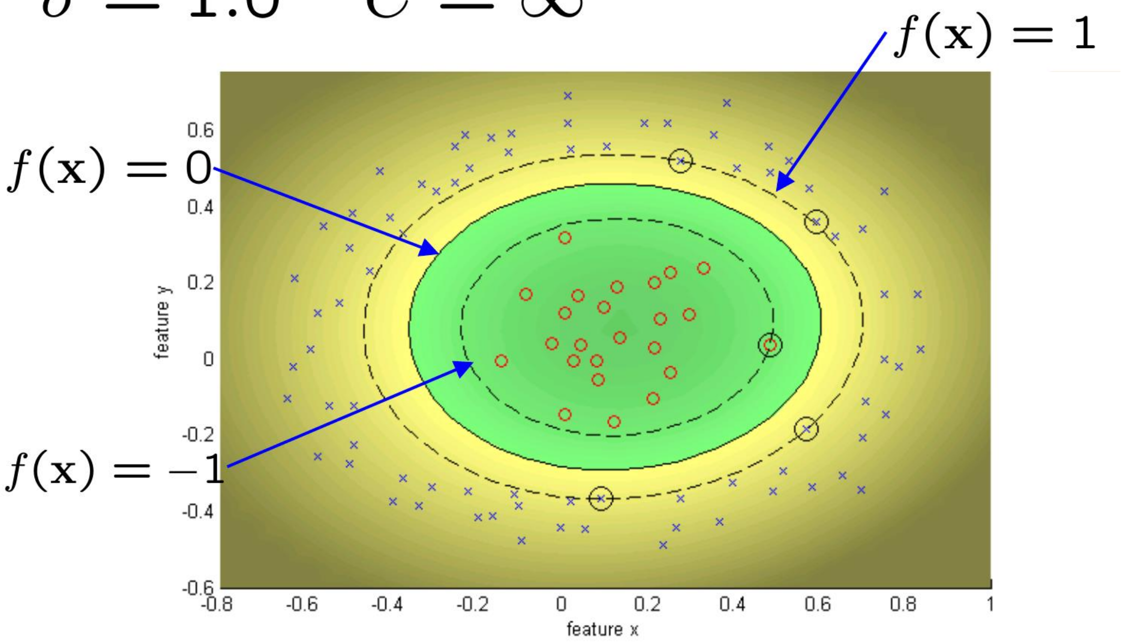
$$k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{2\sigma}\right)$$



Example: SVM with RBF kernel

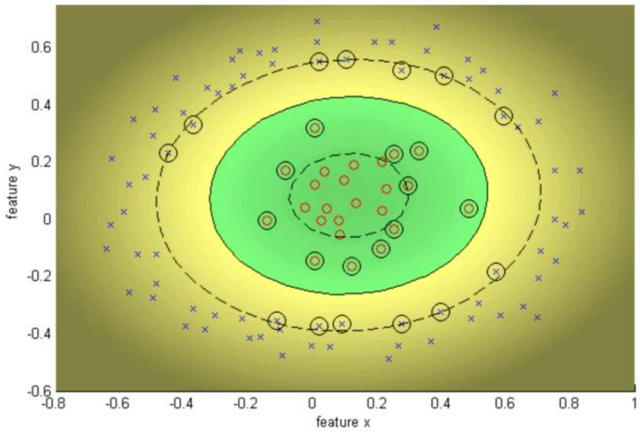
Data is non-linearly separable in feature space

$$\sigma = 1.0 \qquad C = \infty$$

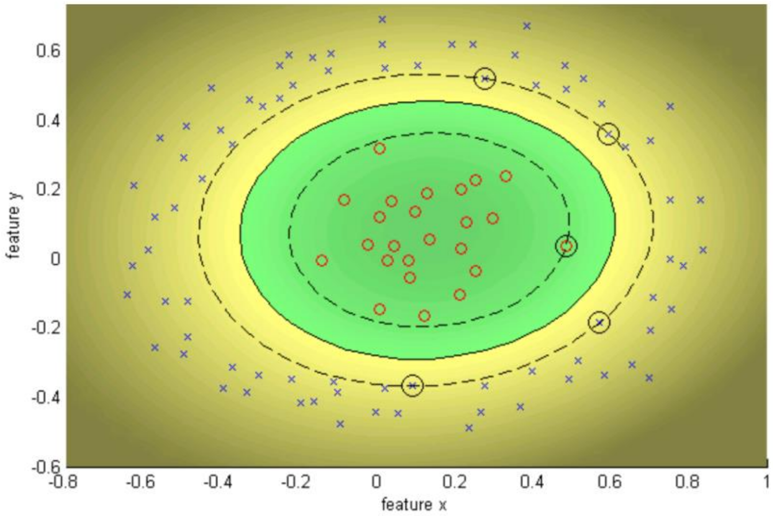


Example: Different Cs

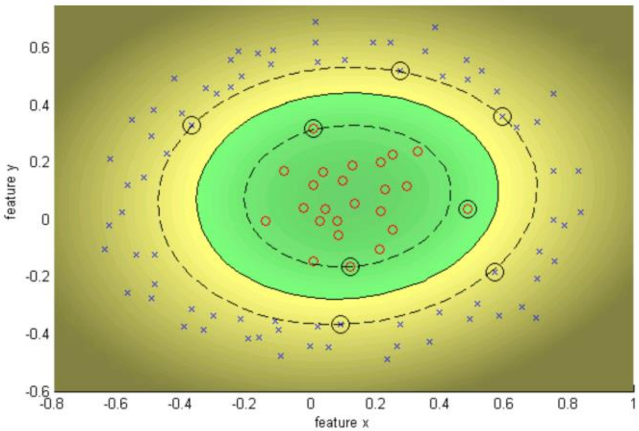
$$\sigma = 1.0 \quad C = 10$$



$$\sigma = 1.0 \quad C = \infty$$

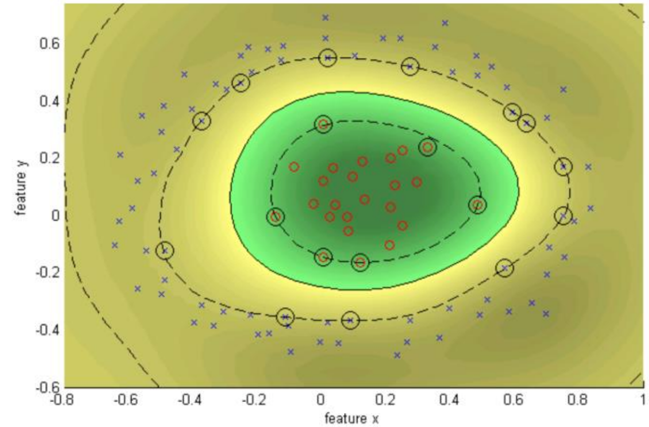


$$\sigma = 1.0 \quad C = 100$$

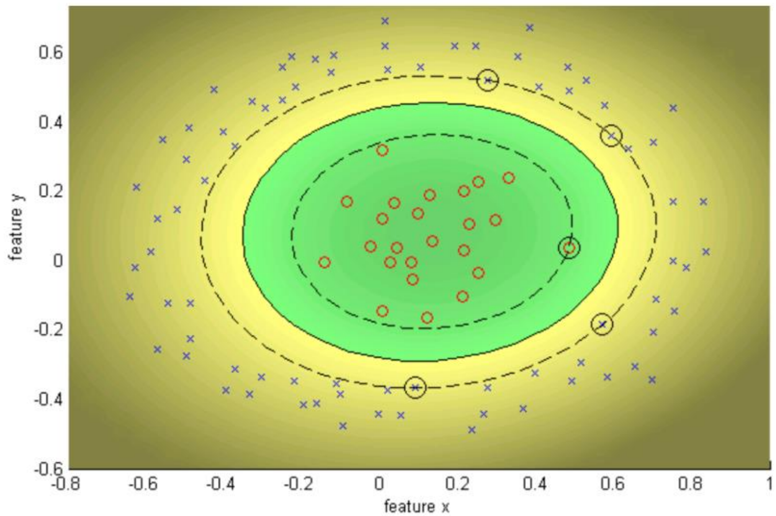


Example: Different sigma

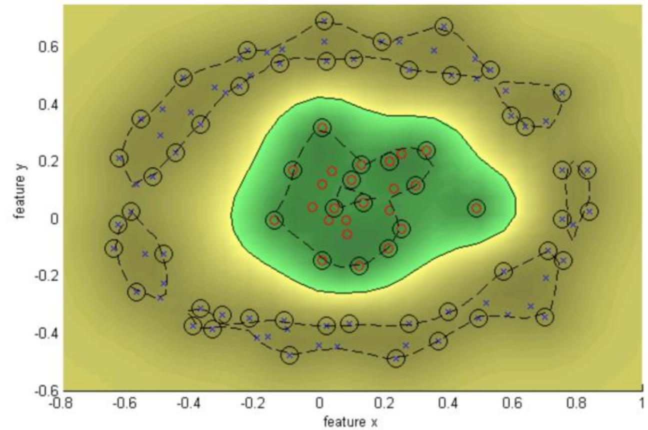
$\sigma = 0.25 \quad C = \infty$



$\sigma = 1.0 \quad C = \infty$



$\sigma = 0.1 \quad C = \infty$



Overfitting

Huge feature space with kernels: should we worry about overfitting?

- SVM objective seeks a solution with large margin
- Theory says that large margin leads to good generalization
- But everything overfits sometimes!!!

Can control overfitting by:

- Setting C (low C -> smaller Complexity)
- Choosing a better Kernel
- Varying parameters of the Kernel (width of Gaussian, etc.)

Model Selection Problem

Handwritten Digit Clasification

US postal service database

- Human performance: 2.5% error

Various learning algorithms (pre-deep learning)

- 16.2%: Decision tree (C4.5)
- 5.9%: 2-layer neural network
- 5.1%: LeNet 1 - 5-layer neural network

Various SVM results

- 4.0%: Polynomial kernel (274 support vectors)
- 4.1%: Gaussian kernel

Handwritten Digit Clasification

Very little overfitting due to max-margin

degree of polynomial	dimensionality of feature space	support vectors	raw error
1	256	282	8.9
2	≈ 33000	227	4.7
3	$\approx 1 \times 10^6$	274	4.0
4	$\approx 1 \times 10^9$	321	4.2
5	$\approx 1 \times 10^{12}$	374	4.3
6	$\approx 1 \times 10^{14}$	377	4.5
7	$\approx 1 \times 10^{16}$	422	4.5

Recent results

- With more training data, better modeling of invariances, etc.
- Error down to about 0.5% with SVMs and 0.4% with neural networks

Takeaway messages

What have we learned today?

Maximum Margin Classifiers:

- A robust formulation for classification
- Margin can be expressed as constrained optimization
- Slack variables allow for constrained violation and regularization
- Can be efficiently optimized using hinge-loss and sub-gradient descent

Sub-gradients:

- Use it for non-differentiable convex functions

Kernel trick for SVMs:

- Results from the Lagrangian dual formulation
- Optimal solution for w is a linear combination of the support vectors (compare to kernel regression)

Self-test questions

You should understand now:

- Why is it good to use a maximum margin objective for classification?
- How can we define the margin as optimization problem?
- What are slack variables and how can they be used to get a “soft” margin?
- How is the hinge loss defined?
- What is the relation between the slack variables and the hinge loss?
- What are the advantages and disadvantages in comparison to logistic regression?
- What is the difference between gradients and sub-gradients