Universität Stuttgart
KI – Institute for Artificial Intelligence

Analytic Computing

Machine Learning
10 Support Vector Machines

Prof. Dr. Steffen Staab

Nadeen Fatallah
Daniel Frank
Akram Sadat Hosseini
Jiaxin Pan

Osama Mohamed
Arvindh Arunbabu
Tim Schneider
Yi Wang

https://www.ki.uni-stuttgart.de/

- based on slides by
  - Thomas Gottron, U. Koblenz-Landau, https://west.uni-koblenz.de/de/studying/courses/ws1718/machine-learning-and-data-mining-1
  - Andrew Zisserman, http://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

1 Perceptron Algorithm

Binary classification

- Given training data $\{(x_i, y_i)\}_{i=1}^{N}$ with $x_i \in \mathbb{R}^d$ and $y_i \in \{-1,1\}$,

- Learn a classifier

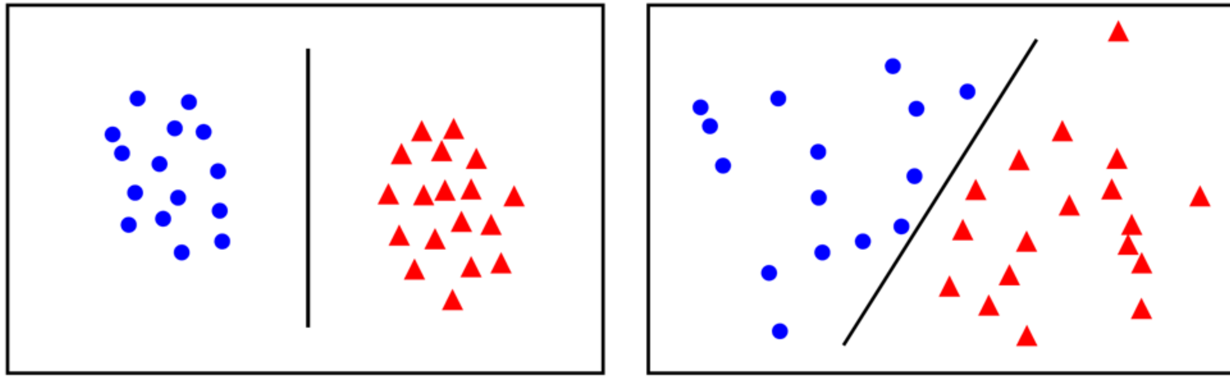
$$\hat{f}(x_i) = \begin{cases} > 0, \text{if } y_i = +1 \\ < 0, \text{if } y_i = -1 \end{cases}$$

- Correct classification:

$$\hat{f}(x_i) y_i > 0$$

Linear separability

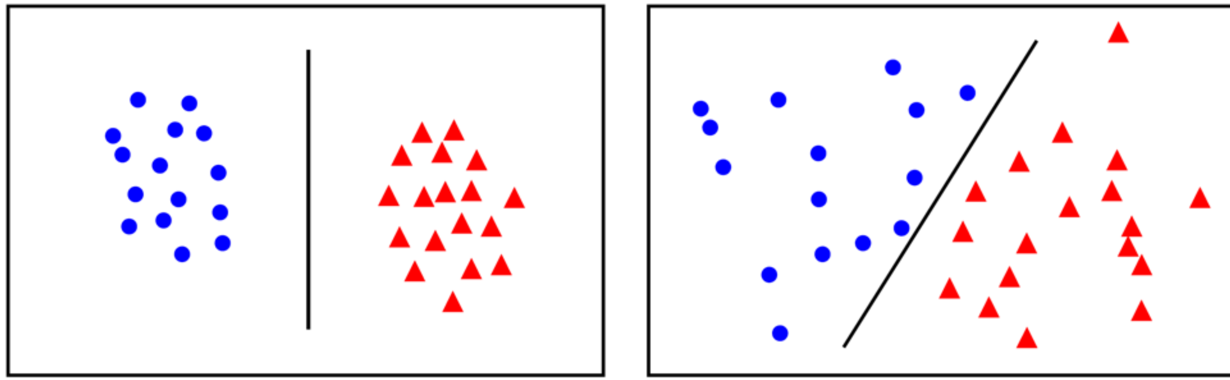
linearly separable



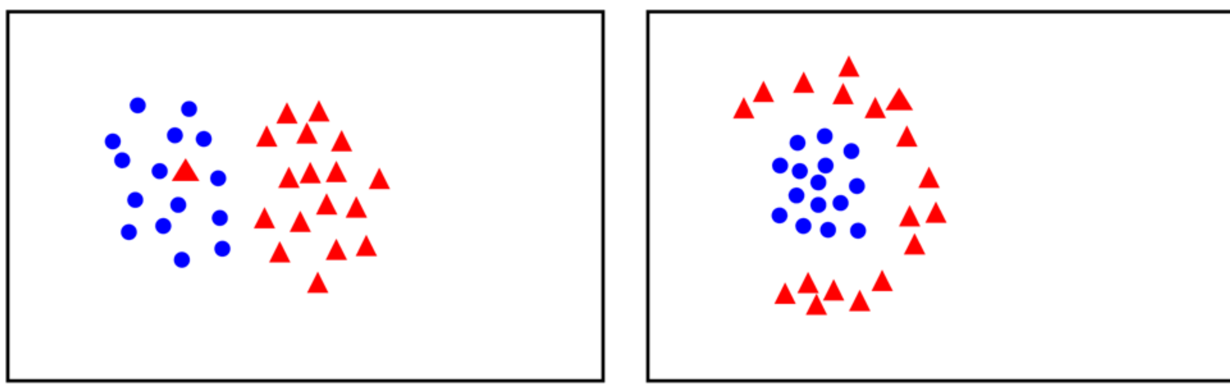
(Zisserman 2015)

Linear separability

linearly
separable



not
linearly
separable



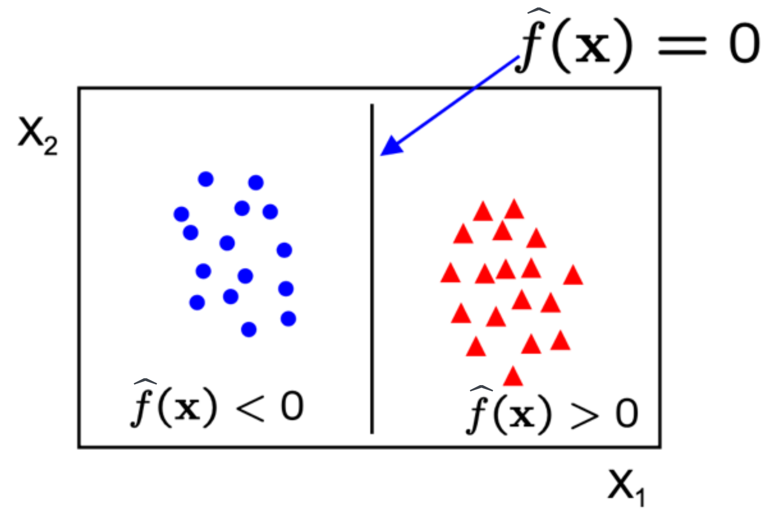
(Zisserman 2015)

Linear classifiers

- A linear classifier has the form

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

- In 2D the discriminant is a line
- w is the normal to the line, and b the bias
- w is known as the weight vector

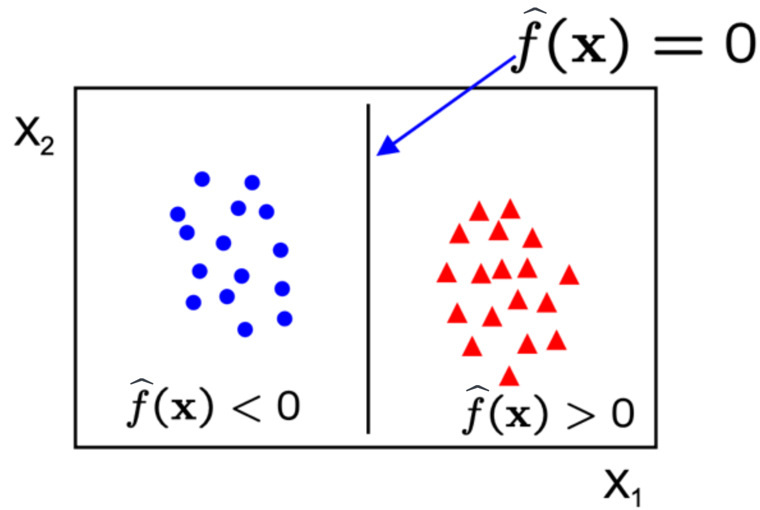


(Zisserman 2015)

Linear classifiers

- Let‘s assume $\boldsymbol{x_i = (1, x_{i,1}, \dots, x_{i,d})}$
  (as in linear regression)
- The we write

$$\hat{f}(x) = w^T x$$



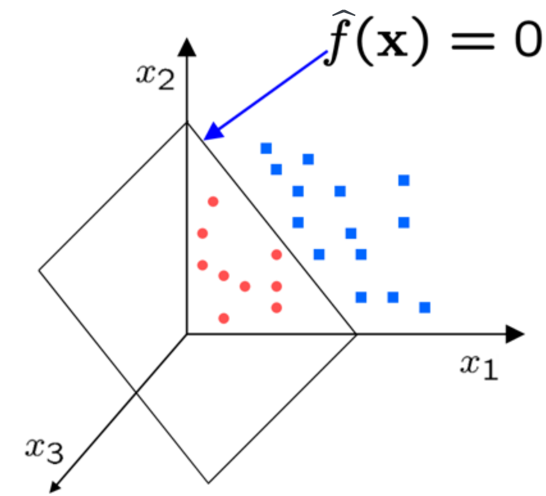
(Zisserman 2015) $x_0=1$

Linear classifiers

- A linear classifier has the form

$$\hat{f}(x) = w^T x$$

- In 3D the discriminant is a plane
- In n-dim the discriminant is a hyperplane
- Only w (including b) are needed to classify new data



(Zisserman 2015)

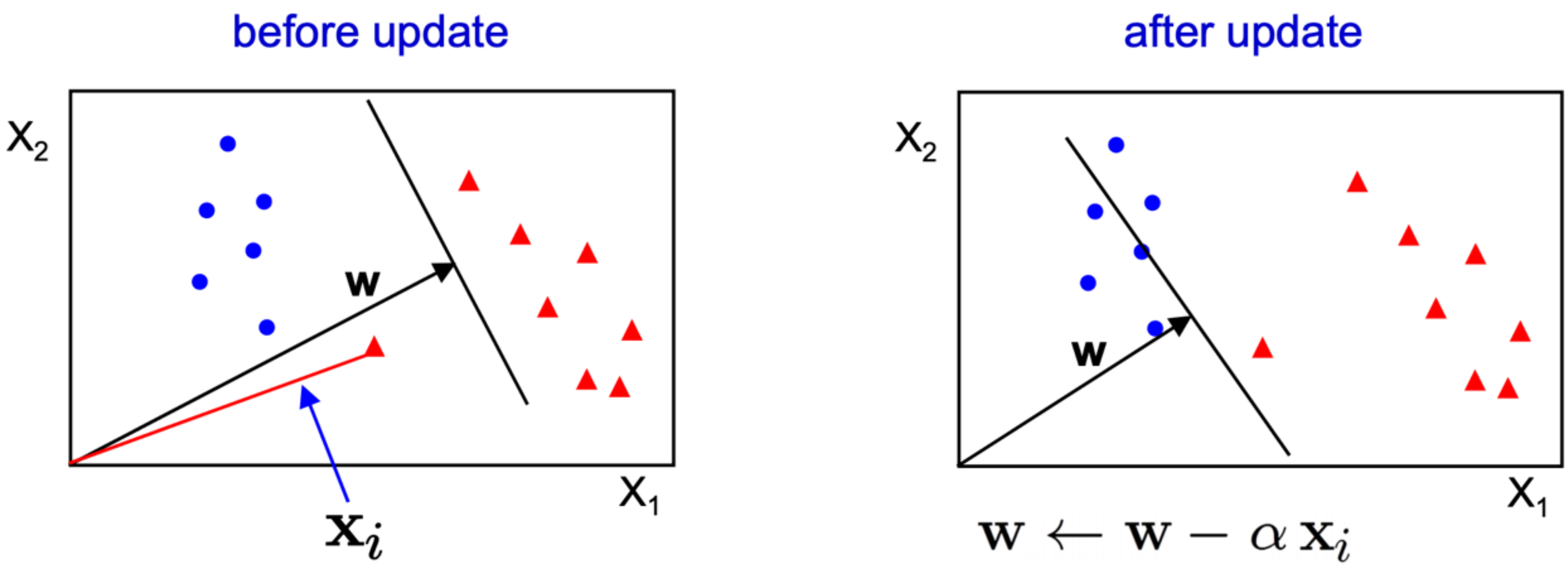
The perceptron classifier

- Given training data $\{(x_i, y_i)\}_{i=1}^{N}$ with $x_i \in \mathbb{R}^d$ and $y_i \in \{-1,1\}$,
- how to find a weight vector w, the *separating hyperplane*, such that the two categories are separated for the dataset?

- Perceptron algorithm
  1. Initialize $w = \bar{0}$
  2. While there is i such that $\hat{f}(x_i)y_i < 0$ do
     - $w \coloneqq w - \alpha x_i \operatorname{sign}\left(\hat{f}(x_i)\right) = w + \alpha x_i y_i$ (not for sign(...)=0)

Example for perceptron algorithm in 2D

1. Initialize $w = \bar{0}$
2. While there is i such that $\hat{f}(x_i)y_i < 0$ do
   - $w := w + \alpha x_i \operatorname{sign}(\hat{f}(x_i))$



At convergence: $w = \sum_{i=1}^{N} \alpha_i \, x_i$

(Zisserman 2015)

Example

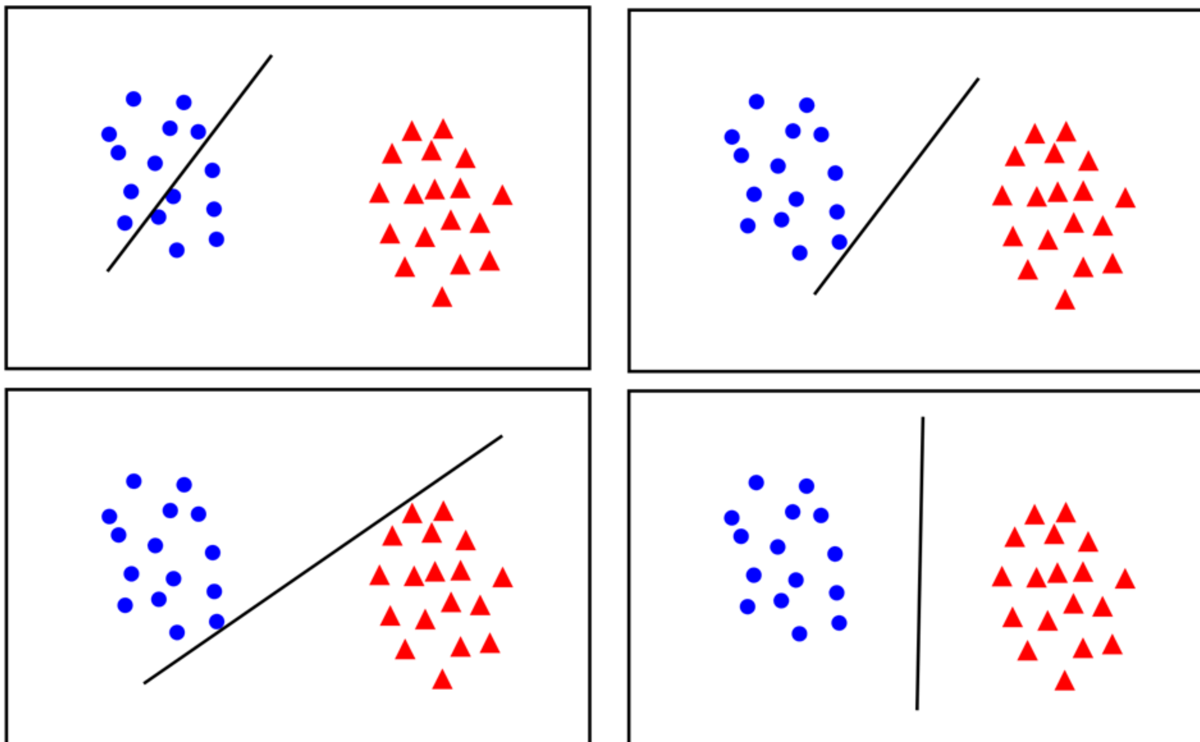
- if the data is linearly separable, then the algorithm will converge
- convergence can be slow …
- separating line close to training data

(Zisserman 2015)

2 Support Vectors

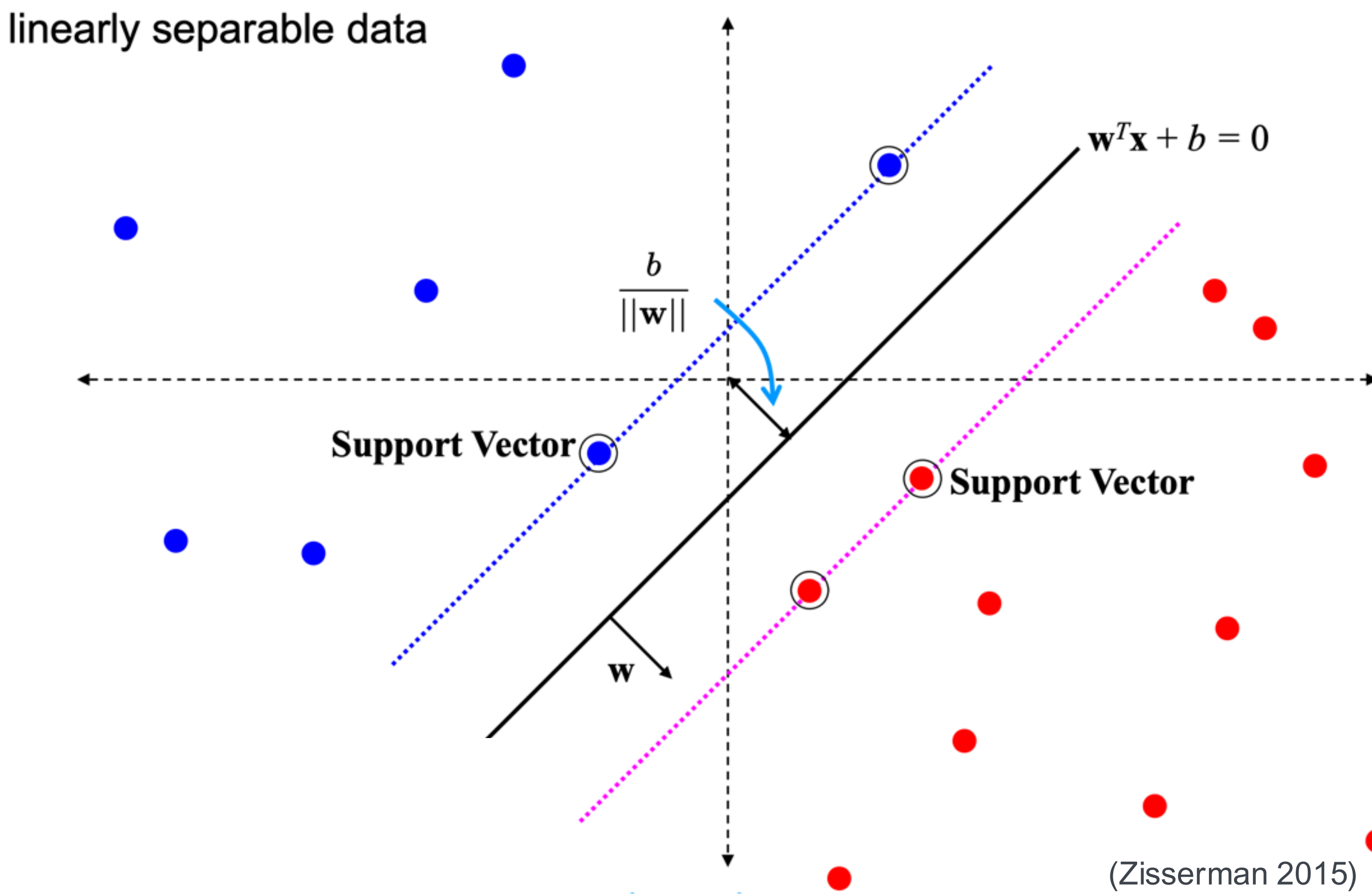
What is the best w?

- Idea: maximum margin solution is most stable under perturbations of the inputs



(Zisserman 2015)

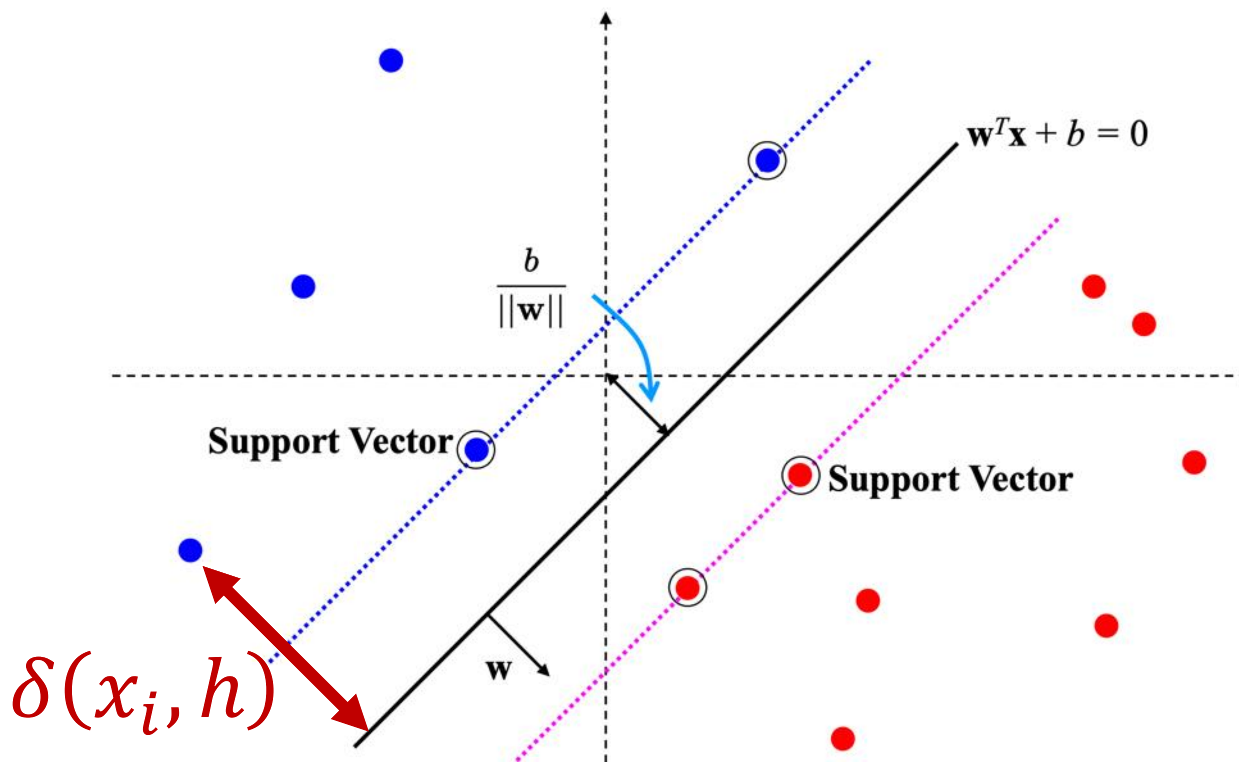
Support Vector Machine



SVM Optimization Problem (1)

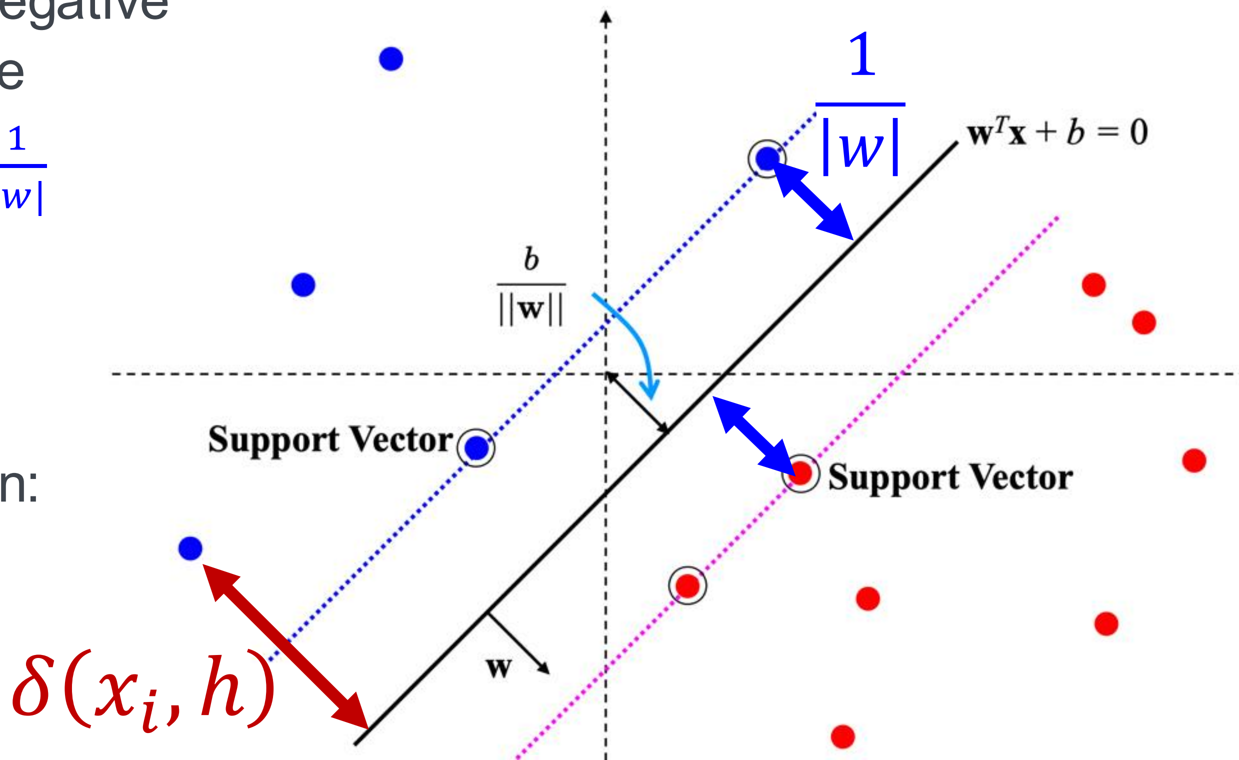
- Distance $\delta(x_i, h)$ of data point x_i from hyperplane h

$$\delta(x_i, h) = \frac{y_i \cdot (w^T x_i + b)}{|w|}$$

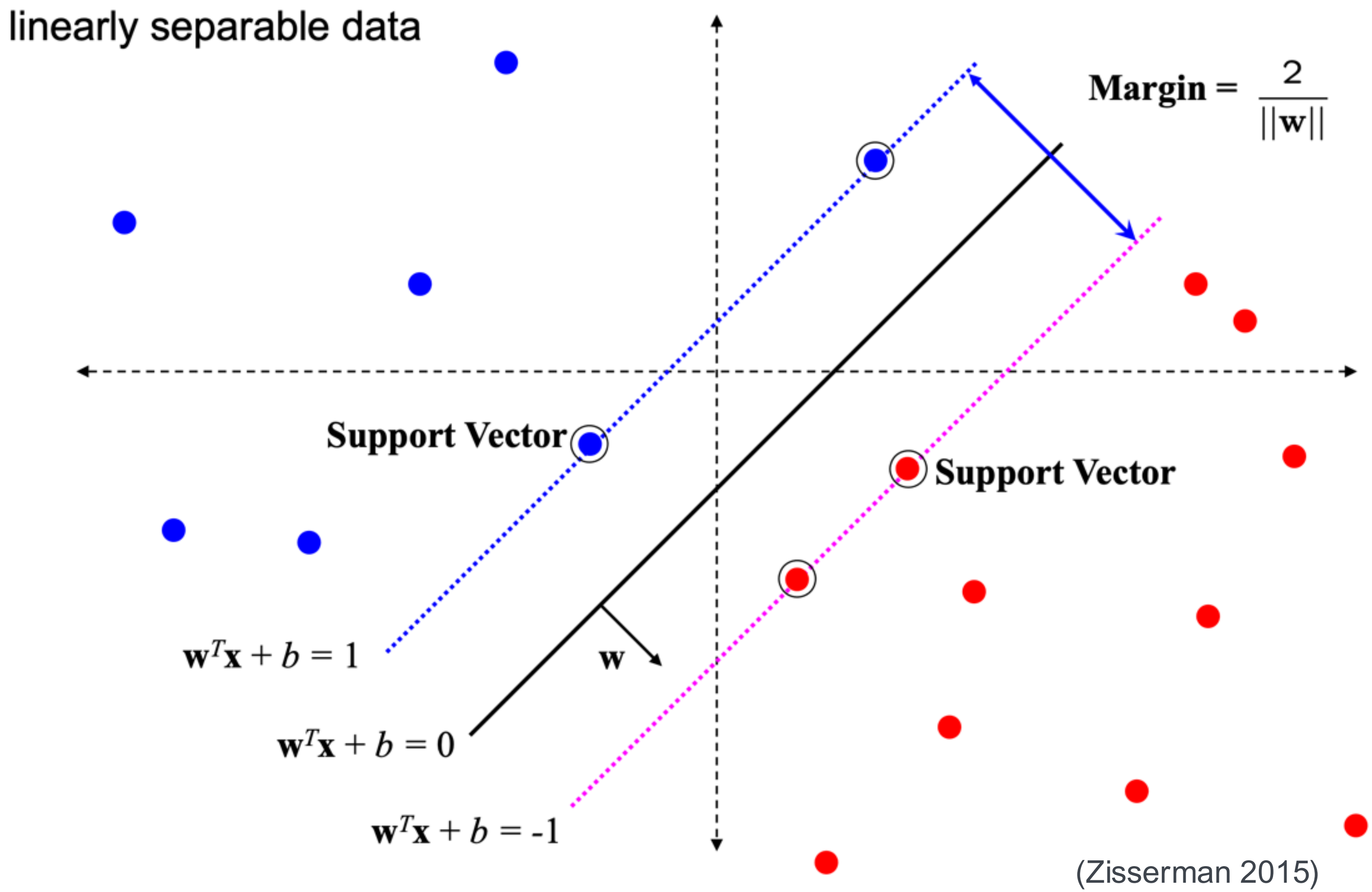


SVM Optimization Problem (1)

- In general, length of vector w does not matter
  - fix w such that support vectors x_j make $y_j \cdot (w^T x_j + b) = 1$
- Then positive and negative support vectors have distance $\delta(x_j, h) = \frac{1}{|w|}$ from hyperplane – which we want to maximize
- Standard formulation: Minimize $\frac{|w|}{2}$ (inverse margin)



Support Vector Machine



SVM Optimization Problem (2)

$$\max_{w} \frac{2}{||w||}$$

subject to

$$y_i\left(w^T x_i + b\right) \geq 1, \text{for } i = 1 \dots N$$

Or equivalently

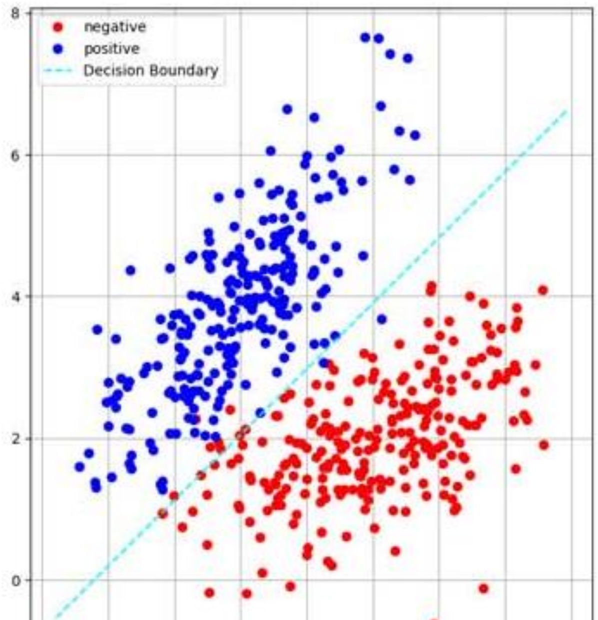
$$\min_{w} ||w||^2$$

subject to the same constraints

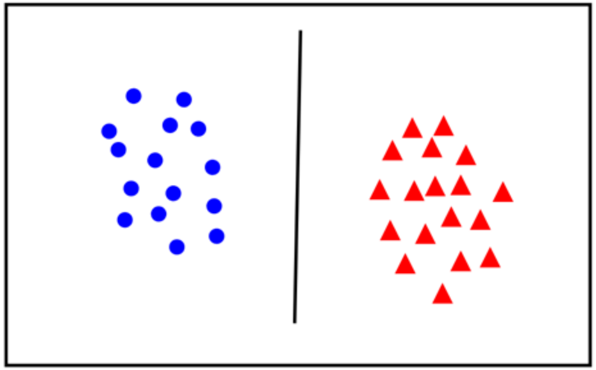
This is a quadratic optimization problem
subject to linear constraints and there is a unique minimum

Compare the two optimization criteria for classfication of linearly separable data by linear regression with classification by SVM

Linear classification using regression: decision line is average between regression lines; all data points are considered



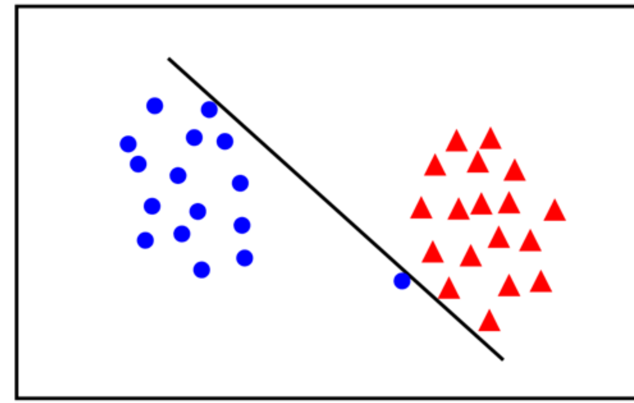
Linear classification using SVM: decision line maximizes margins between support vectors; far away data points are irrelevant



3 Soft Margin and Hinge Loss

Re-visiting linear separability

- Points can be linearly separated, but with very narrow margin



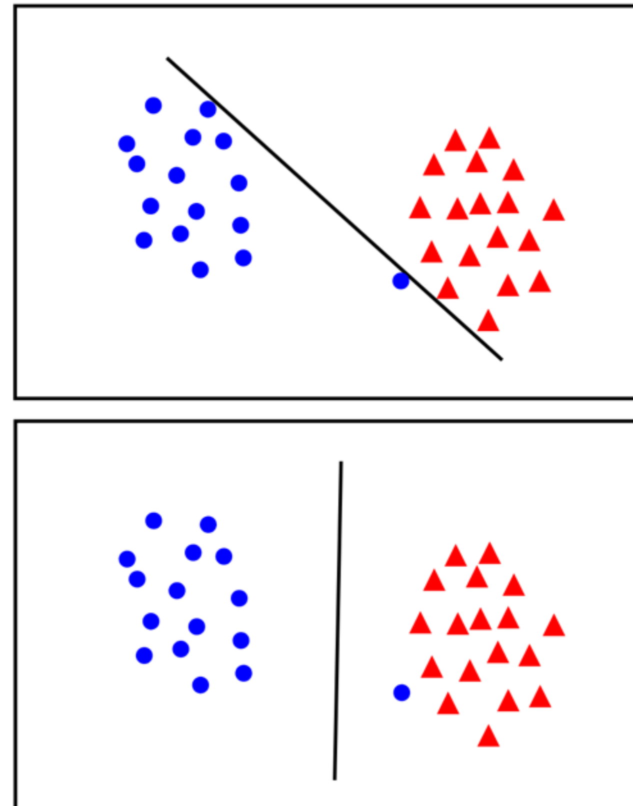
(Zisserman 2015)

Re-visiting linear separability

- Points can be linearly separated, but with very narrow margin

- Possibly the large margin solution is better, even though one constraint is violated

Trade-off between the margin and the number of mistakes on training data

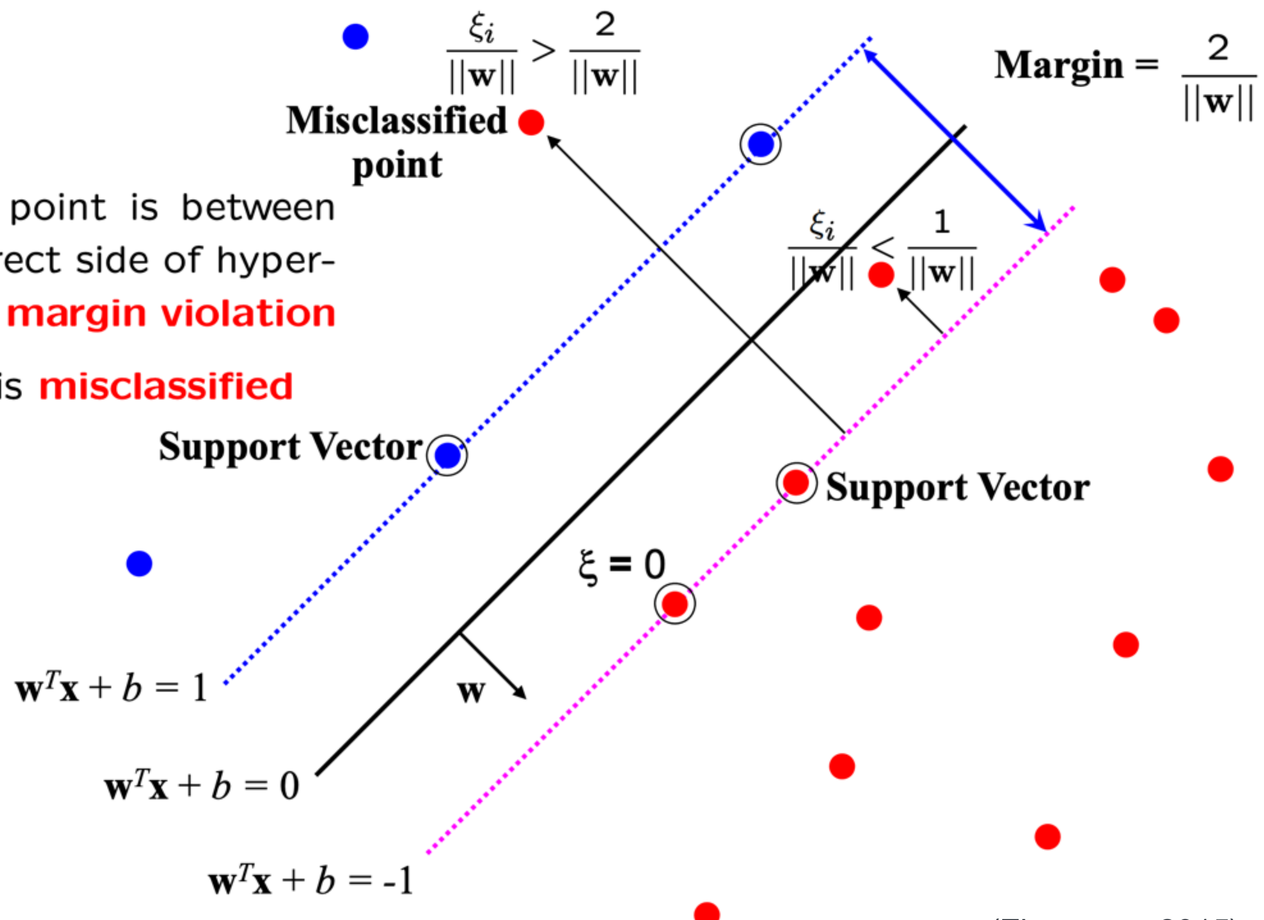


(Zisserman 2015)

Introduce „slack“ variables

$$\xi_i \geq 0$$

- for $0 < \xi \leq 1$ point is between margin and correct side of hyper-plane. This is a **margin violation**
- for $\xi > 1$ point is **misclassified**



(Zisserman 2015)

Soft margin solution

Revised optimization problem

$$\min_{w \in \mathbb{R}^d, \xi_i \in \mathbb{R}^+} ||w||^2 + C \sum_{i=1}^{N} \xi_i$$

subject to

$$y_i\left(w^T x_i + b\right) \geq 1 - \xi_i, \text{for } i = 1 \dots N$$

- Every constraint can be satisfied if ξ_i is sufficiently large
- C is a regularization parameter:
  - small C allows constraints to be easily ignored $\Longrightarrow$ large margin
  - large C makes constraints hard to ignore $\Longrightarrow$ narrow margin
  - $C = \infty$ enforces all constraints $\Longrightarrow$ hard margin
- Still a quadratic optimization problem with unique minimum
- One hyperparameter C

Loss function

- Given constraints:

$$y_i\left(w^T x_i + b\right) \geq 1 - \xi_i$$
$$\xi_i \geq 0$$

- We can rewrite ξ_i as:

$$\xi_i = \max(0,1 - y_i \hat{f}(x_i))$$

- Hence, we can optimize the unconstrained optimization problem over w:

$$\min_{w \in \mathbb{R}^d} \underbrace{\frac{1}{C}\|w\|^2}_{\text{regularization}} + \sum_{i=1}^{N} \underbrace{\max(0,1 - y_i \hat{f}(x_i))}_{\text{loss function}}$$

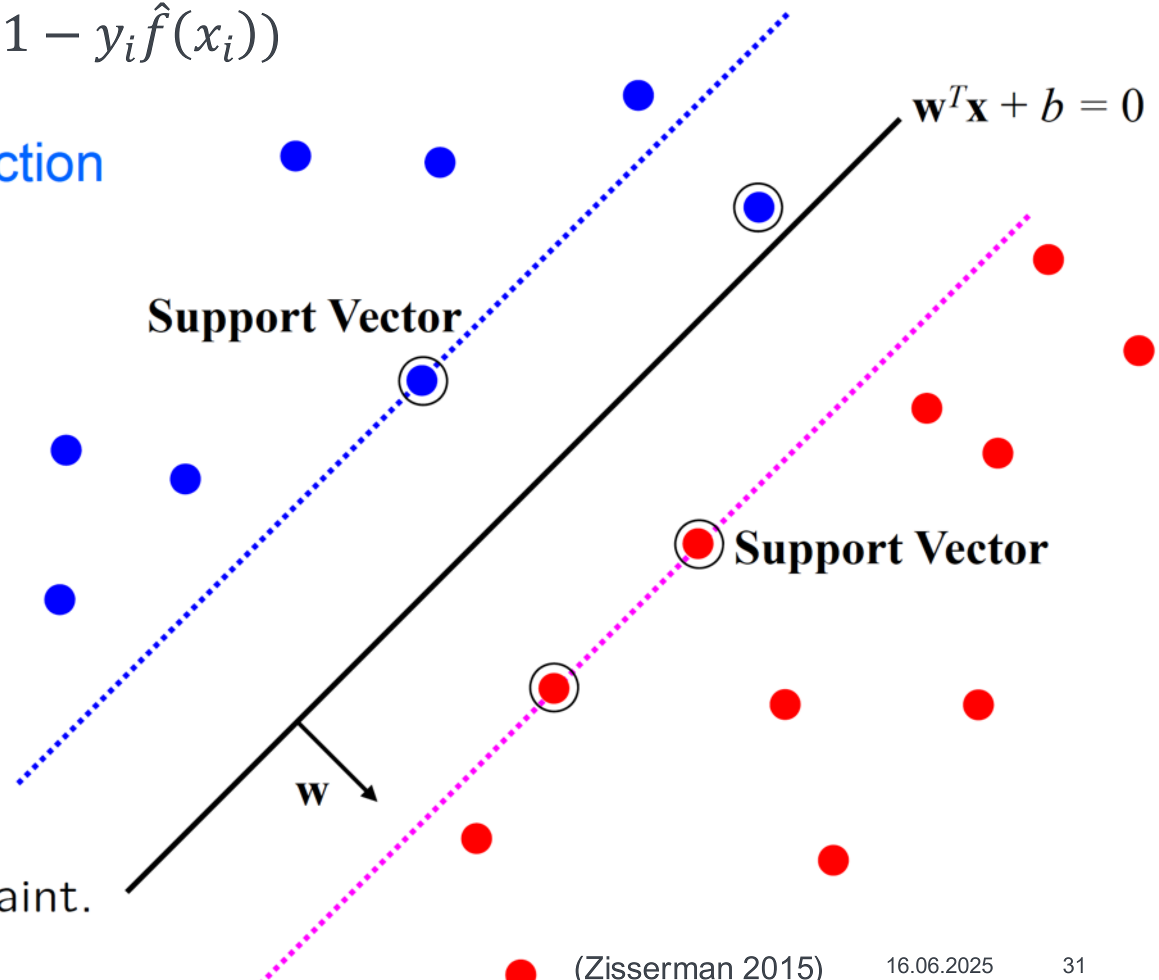
Loss function

$$\min_{w \in \mathbb{R}^d, \xi_i \in \mathbb{R}^+} \frac{1}{C} ||w||^2 + \sum_{i=1}^{N} \max(0, 1 - y_i \hat{f}(x_i))$$

loss function

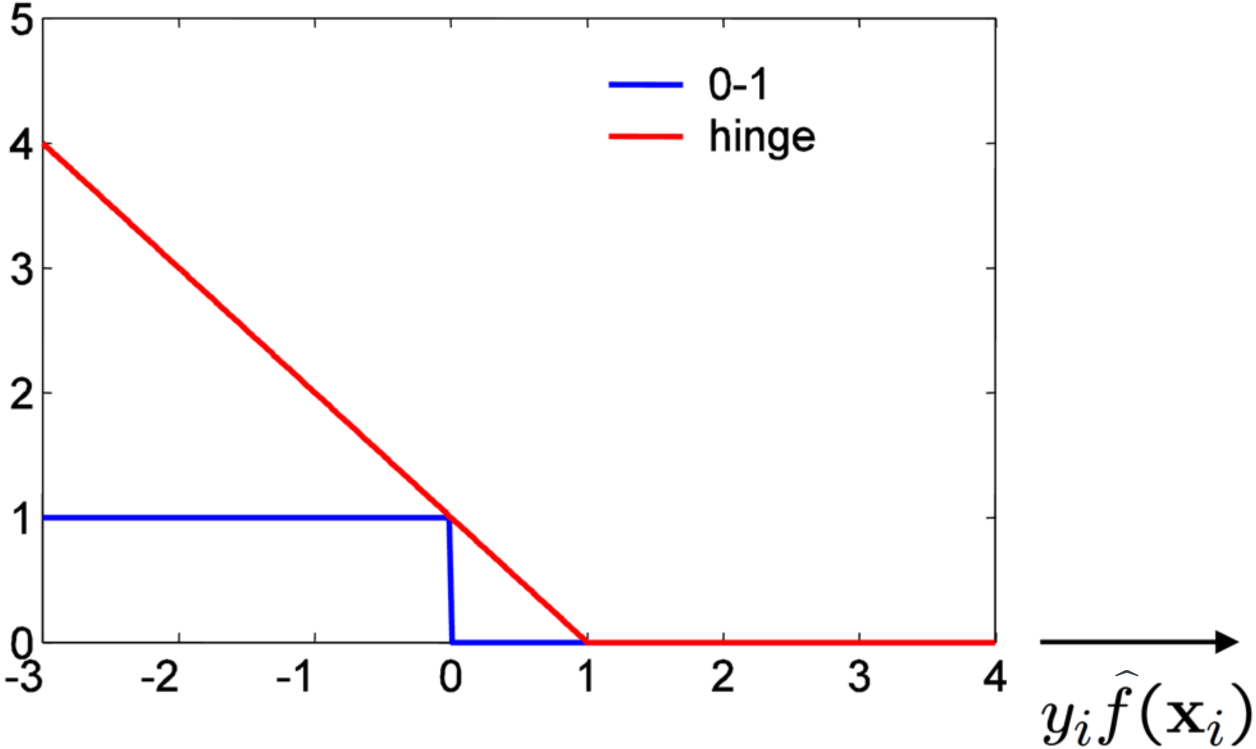
Points are in three categories:

1. $y_i f(x_i) > 1$
   Point is outside margin.
   No contribution to loss
2. $y_i f(x_i) = 1$
   Point is on margin.
   No contribution to loss.
   As in hard margin case.
3. $y_i f(x_i) < 1$
   Point violates margin constraint.
   Contributes to loss



(Zisserman 2015)

Hinge loss



- SVM uses "hinge" loss $\max\left(0, 1 - y_i \widehat{f}(\mathbf{x}_i)\right)$
- an approximation to the 0-1 loss

(Zisserman 2015)

4 Gradient descent over convex function

Gradient descent/ascent

Climb down a hill
Climb up a hill

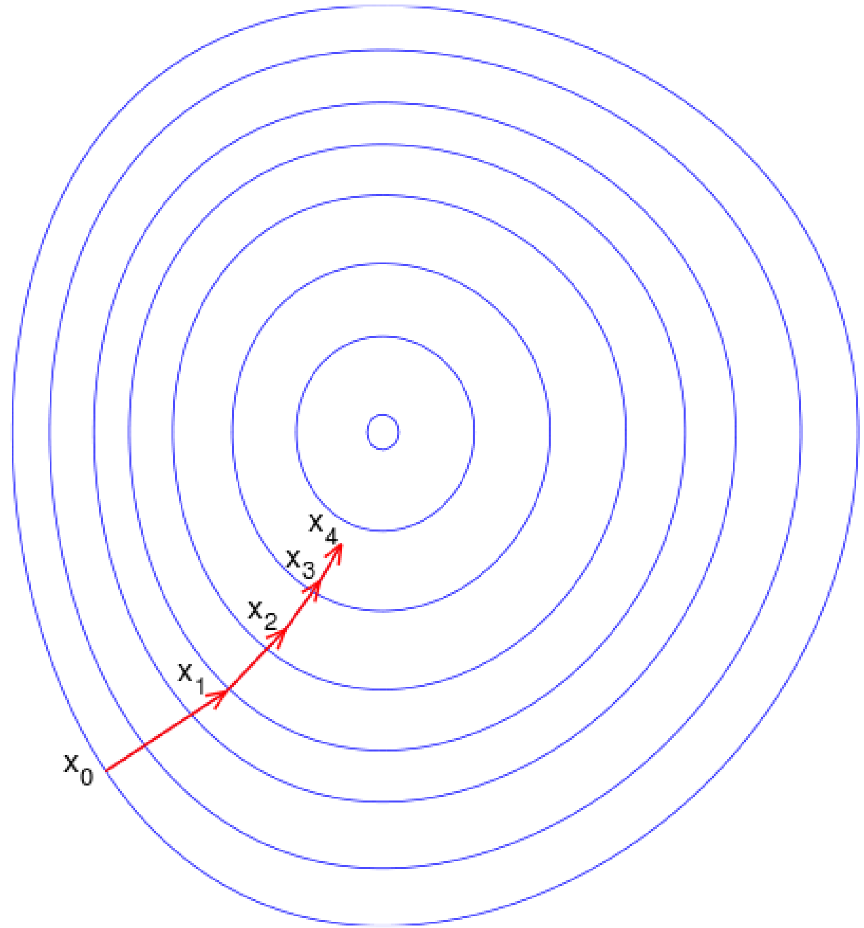
Given differentiable function describing height of hill at position $\boldsymbol{x} = (x_1, \dots, x_k)$ height of hill $f(\boldsymbol{x})$.

How to climb up/down fastest?

Go in direction where

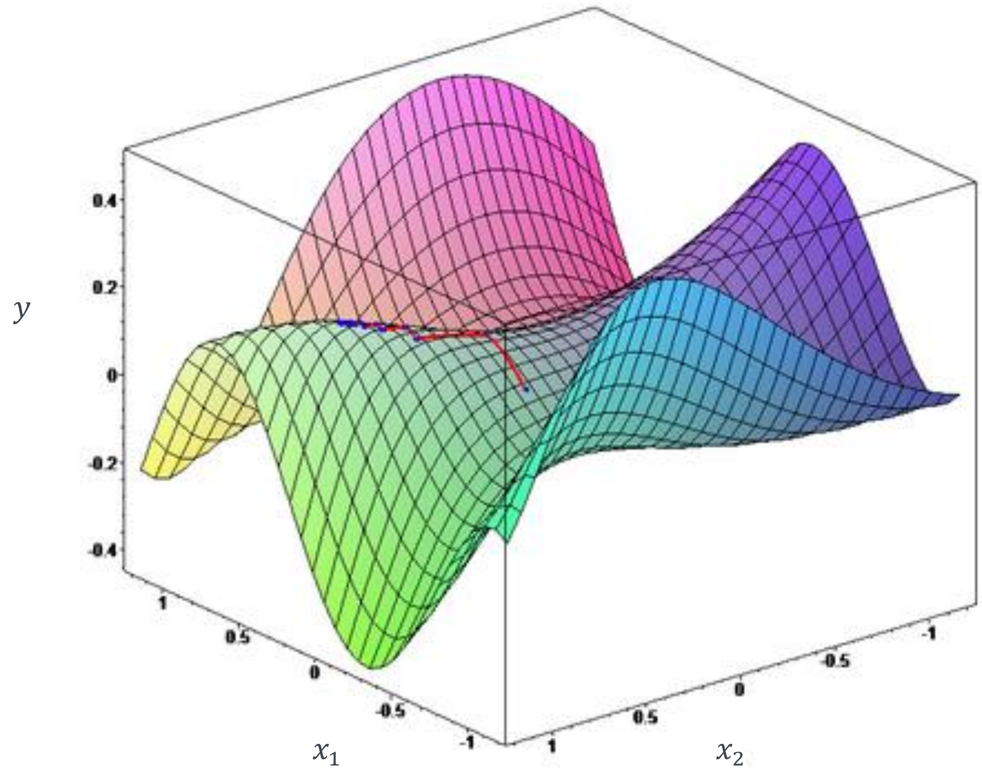
$$\frac{df(\boldsymbol{x})}{d\boldsymbol{x}} = \nabla_{\boldsymbol{x}} f(\boldsymbol{x})$$

is maximal/minimal



https://en.wikipedia.org/wiki/Gradient_descent#/media/File:Gradient_descent.svg

In general: challenge can be difficult



Gradient Descent (- but without posts)

https://goo.gl/images/JKN6zm

Optimization continued

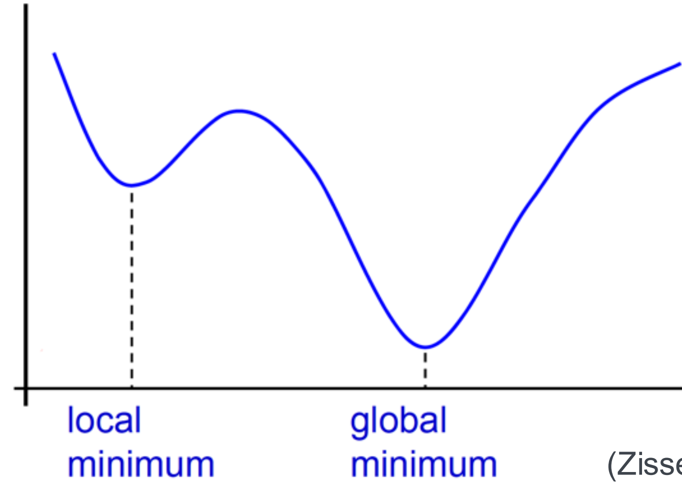
$$\min_{\mathbf{w}\in\mathbb{R}^d} C\sum_{i}^{N} \max\left(0, 1 - y_i\widehat{f}(\mathbf{x}_i)\right) + ||\mathbf{w}||^2$$

Questions

- Does this cost function have a unique solution?

Optimization continued

$$\min_{\mathbf{w}\in\mathbb{R}^d} C\sum_{i}^{N} \max\left(0, 1 - y_i f(\mathbf{x}_i)\right) + ||\mathbf{w}||^2$$



(Zisserman 2015)

Questions

- Does this cost function have a unique solution?
- Do we find it using gradient descent?
  Does the solution we find using gradient descent depend on the starting point?

To the rescue:

- If the cost function is convex, then a locally optimal point is globally optimal (provided the optimization is over constraints that form a convex set – given in our case)

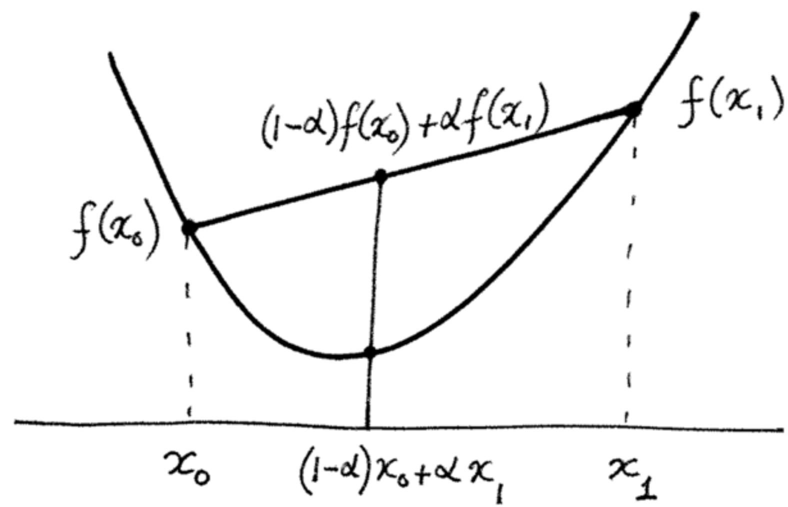
Convex functions

D – a domain in $\mathbb{R}^n$.

A convex function $f : D \to \mathbb{R}$ is one that satisfies, for any $\mathbf{x}_0$ and $\mathbf{x}_1$ in D:

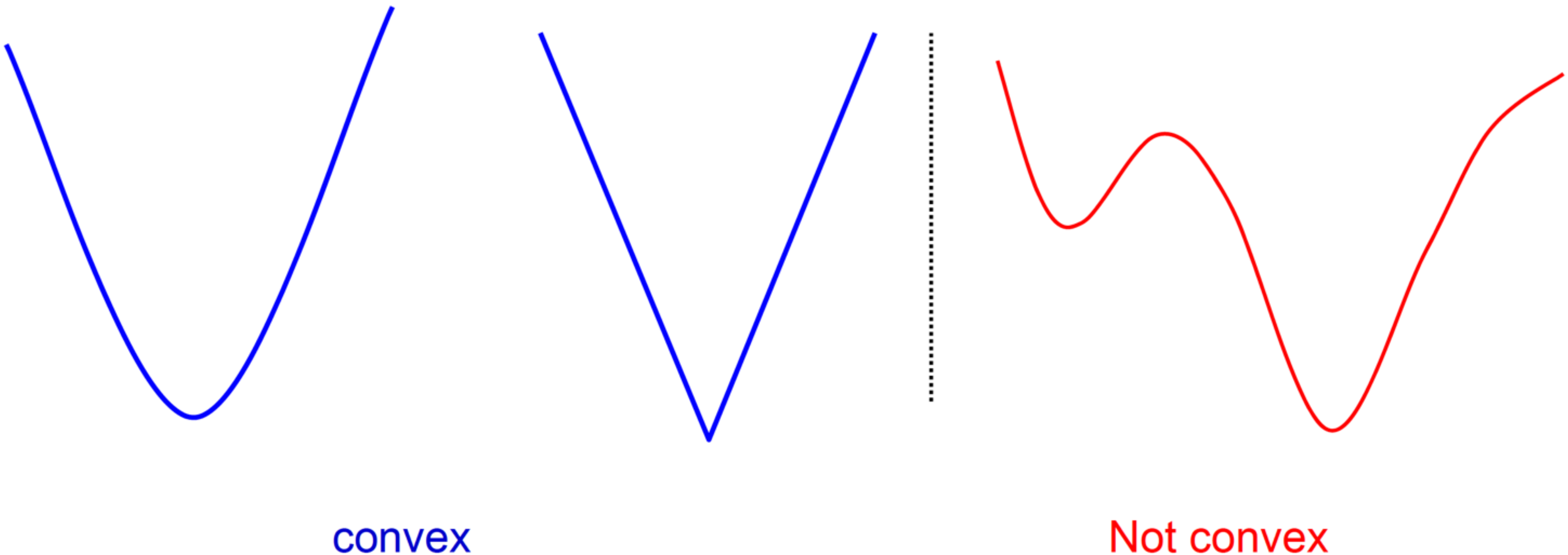
$$f((1-\alpha)\mathbf{x}_0 + \alpha\mathbf{x}_1) \leq (1-\alpha)f(\mathbf{x}_0) + \alpha f(\mathbf{x}_1) \quad .$$

Line joining $(\mathbf{x}_0, f(\mathbf{x}_0))$ and $(\mathbf{x}_1, f(\mathbf{x}_1))$ lies above the function graph.



(Zisserman 2015)

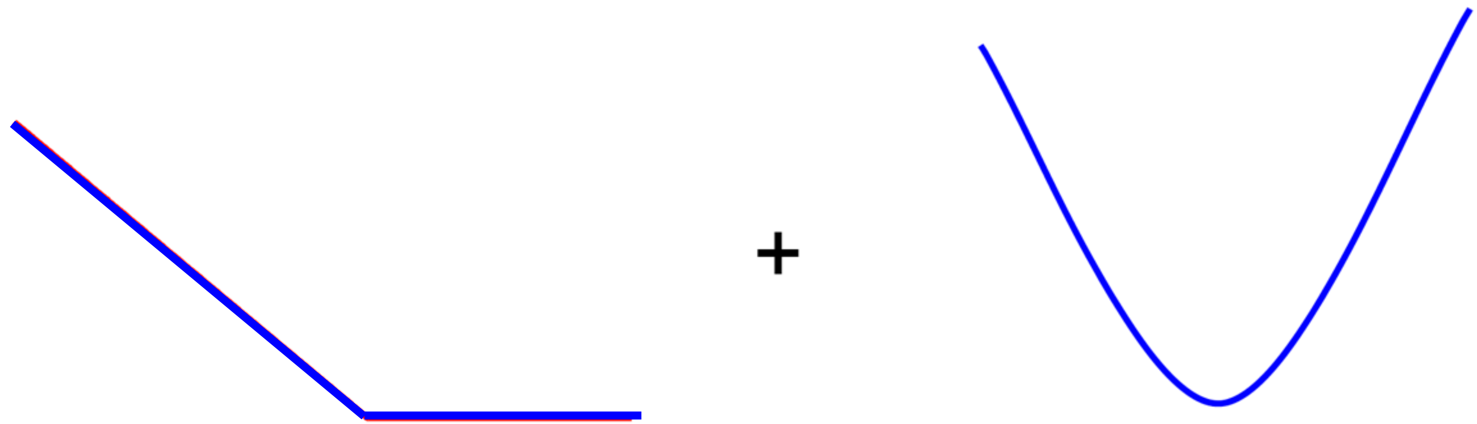
Convex function examples



- A non-negative sum of convex functions is convex

(Zisserman 2015)

Applied to hinge loss and regularization



SVM

$$\min_{\mathbf{w}\in\mathbb{R}^d} C \sum_{i}^{N} \max\left(0, 1 - y_i f(\mathbf{x}_i)\right) + ||\mathbf{w}||^2$$

convex

Gradient descent algorithm for SVM

To minimize a cost function $\mathcal{C}(w)$ use the iterative update

$$w_{t+1} \coloneqq w_t - \eta_t \nabla_{\mathrm{w}} \mathcal{C}(\mathrm{w_t})$$

where η is the learning rate.

Let‘s rewrite the minimization problem as an average with $\lambda = \frac{2}{C}$:

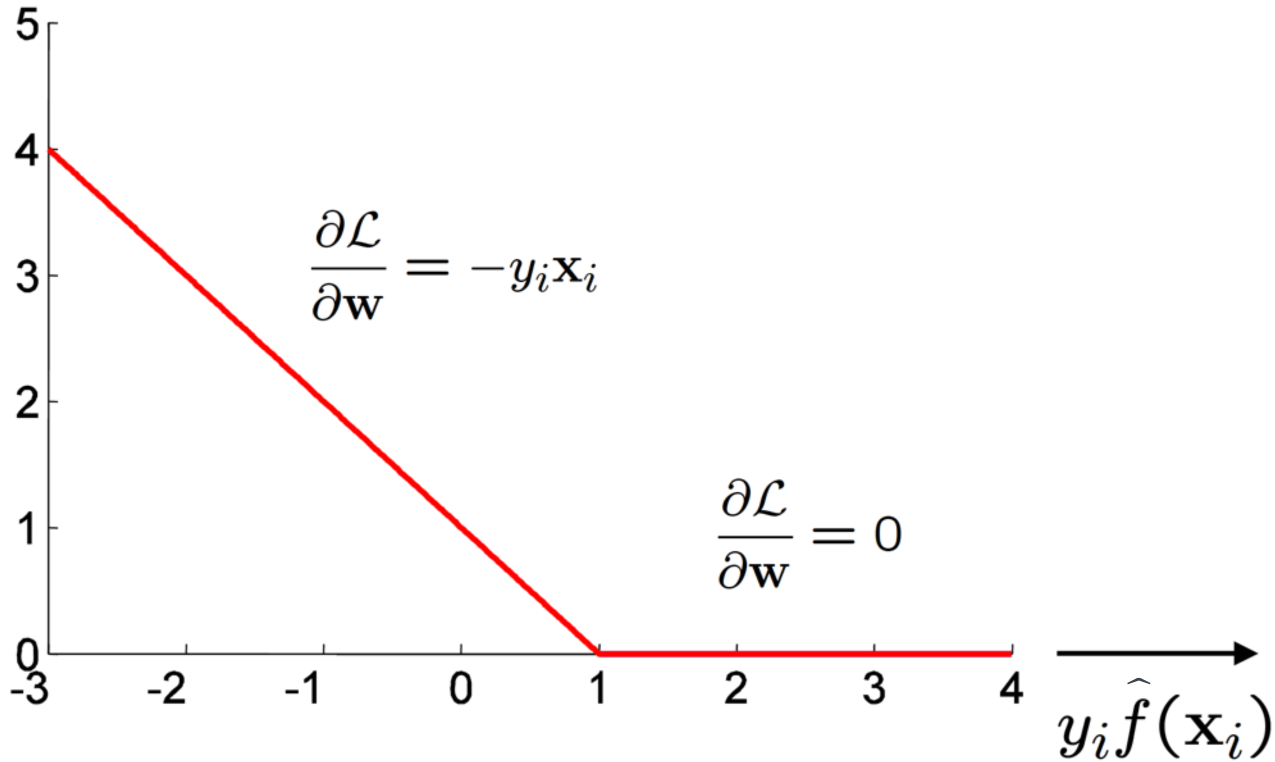
$$\mathcal{C}(w) = \frac{1}{NC} \|w\|^2 + \frac{1}{N} \sum_{i=1}^{N} \max(0,1 - y_i \hat{f}(x_i)) =$$

$$= \frac{1}{N} \sum_{i=1}^{N} \left(\frac{\lambda}{2} \|w\|^2 + \max(0,1 - y_i \hat{f}(x_i)) \right)$$

and $\hat{f}(x_i) = w^T x + b$

Sub-gradient for hinge loss

$\mathcal{L}(x_i, y_i; w) = \max(0, 1 - y_i \hat{f}(x_i))$, $\hat{f}(x_i) = w^T x_i + b$



(Zisserman 2015)

Sub-gradient descent algorithm for SVM

$$\mathcal{C}(w) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\lambda}{2}\|w\|^2 + \mathcal{L}(x_i, y_i; w)\right)$$

The iterative update is

$$w_{t+1} \coloneqq w_t - \eta \nabla_w \mathcal{C}(\mathrm{w_t}) \coloneqq$$

$$\coloneqq w_t - \eta \frac{1}{N}\sum_{i=1}^{N}(\lambda w_t + \nabla_w \mathcal{L}(x_i, y_i; w))$$

Then each iteration t involves cycling through the training data with the updates:

$$w_{t+1} \coloneqq \begin{cases} w_t - \eta(\lambda w_t - y_i x_i), & \text{if } \; y_i \hat{f}(x_i) < 1 \\ w_t - \eta \lambda w_t, & \text{otherwise} \end{cases}$$

Typical learning rate in Pegasos: $\eta_t = \frac{1}{\lambda t}$

5 The dual problem

Primal vs dual problem

- **SVM** is a linear classifier: $\hat{f}(x) = w^T x + b$
- **The primal problem**: an optimization problem over w:

$$\min_{w \in \mathbb{R}^d} \frac{1}{C} \|w\|^2 + \sum_{i=1}^{N} \max(0, 1 - y_i \hat{f}(x_i))$$

- **The dual problem**: Getting rid of the w for a slightly different representation of $\hat{f}(x)$ leads to the following representation

$$\hat{f}(x) = \sum_{i=1}^{N} \alpha_i \, y_i \left(x_i^T x \right) + b$$

and a new optimization problem with the same solution, but several advantages. Let us show this on following slides...

Revisit Optimization Problem for Hard Margin Case

- Minimize the quadratic form

$$\frac{||w||^2}{2} = \frac{w^T w}{2}$$

  - With constraints

$$y_i \cdot (w^T x_i + b) \geq 1 \;\; \forall i$$

- The constraints will reach a value of 1 for at least one instance.

- Include hard constraints into the loss function:

$$\mathcal{L}(w, b, \alpha) = \frac{||w||^2}{2} - \sum_{i=1}^{N} \alpha_i (y_i \cdot (w^T x_i + b) - 1)$$

- failed constraints “punish“ the objective function

Excursion: Lagrange Multiplier

- We want to maximize a function $f(x)$ under the constraints $g(x) = a$

Solution with Lagrange Multiplier

- Optimize the Lagrangian

$$f(x) - \lambda(g(x) - a)$$

instead!

Nicely visual explanation of Lagrange optimization at
https://www.svm-tutorial.com/2016/09/duality-lagrange-multipliers/

Algorithm for optimization with a Lagrange multiplier

1. Write down the Lagrangian $f(x) - \lambda \cdot (g(x) - a)$
2. Take derivative of Lagrangian wrt x,
   set it to 0
   to find estimate of x that depends on λ
3. Plug your estimate of x in the Lagrangian,
   take the derivative wrt λ,
   and set it to 0,
   to find the optimal value for the lagrange multiplier λ
4. Plug in the Lagrange multiplier in your estimate for x

Revisit Optimization Problem for Hard Margin Case

- Minimize the quadratic form

$$\frac{||w||^2}{2} = \frac{w^T w}{2}$$

  - With constraints

$$y_i \cdot (w^T x_i + b) \geq 1 \;\; \forall i$$

- The constraints will reach a value of 1 for at least one instance.

- Include hard constraints into the loss function:

$$\mathcal{L}(w, b, \alpha) = \frac{||w||^2}{2} - \sum_{i=1}^{N} \alpha_i (y_i \cdot (w^T x_i + b) - 1)$$

α_i are the Lagrange multipliers

  - failed constraints "punish" the objective function

Lagrangian primal problem

- Lagrangian primal problem is:

$$\min_{w,b} \max_{\alpha} \mathcal{L}(w, b, \alpha)$$

subject to $\forall i$: $\alpha_i \geq 0$

Finding the optimum

- Loss is a function of w, b, and α

$$\mathcal{L}(w,b,\alpha) = \frac{\|w\|^2}{2} - \sum_{i=1}^{N} \alpha_i(y_i \cdot (w^T x_i + b) - 1)$$

- Find optimum using derivatives:

w is a linear combination of the data instances!

$$\frac{\partial}{\partial b}\mathcal{L}(w,b,\alpha) = 0 \implies 0 = \sum_{i=1}^{N} \alpha_i y_i$$

$$\frac{\partial}{\partial w_j}\mathcal{L}(w,b,\alpha) = 0 \implies w_j = \sum_{i=1}^{N} \alpha_i y_i x_{i,j} \implies w = \sum_{i=1}^{N} \alpha_i y_i x_i$$

Substitution into $\mathcal{L}(w, b, \alpha)$

$$\mathcal{L}(w,b,\alpha)_{|w=\sum_{i=1}^{N}\alpha_i y_i x_i} = \frac{\|w\|^2}{2} - \sum_{i=1}^{N} \alpha_i \left(y_i \cdot \left(w^T x_i + b\right) - 1\right) =$$

$$= \frac{1}{2}\left\{\sum_{j=1}^{N} \alpha_j y_j x_j\right\}^T \left\{\sum_{k=1}^{N} \alpha_k y_k x_k\right\} - \sum_{i=1}^{N} \alpha_i \left(y_i \cdot \left(\left\{\sum_{j=1}^{N} \alpha_j y_j x_j\right\}^{\mathrm{T}} x_i + b\right) - 1 \right) =$$

$$= \underbrace{\frac{1}{2}\sum_{j,k} \alpha_j \alpha_k y_j y_k \left(x_j^T x_k\right) - \sum_{i,j} \alpha_i \alpha_j y_i y_j \left(x_i^T x_j\right)}_{=} - \underbrace{b \sum_{i=1}^{N} \alpha_i y_i}_{=0} + \sum_{i=1}^{N} \alpha_i =$$

$$= \mathcal{L}(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{j,k} \alpha_j \alpha_k y_j y_k \left(x_j^T x_k\right)$$

Wolfe dual problem

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{j,k} \alpha_j \alpha_k y_j y_k \left(x_j^T x_k \right)$$

subject to $\forall i$: $\alpha_i \geq 0$, and $0 = \sum_{i=1}^{N} \alpha_i y_i$

- This problem is solvable with quadratic programming, because it fulfills the Karush-Kuhn-Tucker conditions on α_i that handle inequality constraints (≥1) in the Lagrange optimization (not given here!).
- It gives us the classification function:

$$\hat{f}(x) = \sum_{i=1}^{N} \alpha_i \cdot y_i \cdot x_i^T x + b$$

α_i is positive if x_i is a support vector

Non-separable Case (similar as before)

- Introduce (positive) „slack variables“ ξ_i to allow deviations from the minimum distance:

$$y_i(w^T x_i + b) \geq 1 - \xi_i$$

- Include a penalizing term in the optimization function:

$$C\left(\sum_{i=1}^{m} \xi_i\right)^k$$

- Transform to Lagrangian
  - **with additional Lagrange multipliers** for the slack variables being constrained to positive values ...

Summary: Primal and dual formulations

- **Primal** version of classifier

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

- **Dual** version of classifier

$$\hat{f}(x) = \sum_{i=1}^{N} \alpha_i \cdot y_i \cdot x_i^T x + b$$

The dual form classifier seems to work like a kNN classifier, it requires the training data points x_i. However, many of the α_i are zero.
The ones that are non-zero define the support vectors x_i.

Summary: Primal and dual formulations

- Lagrangian **primal** problem is:

$$\min_{w,b} \max_{\alpha} \mathcal{L}(w, b, \alpha)$$

subject to $\forall i$: $\alpha_i \geq 0$

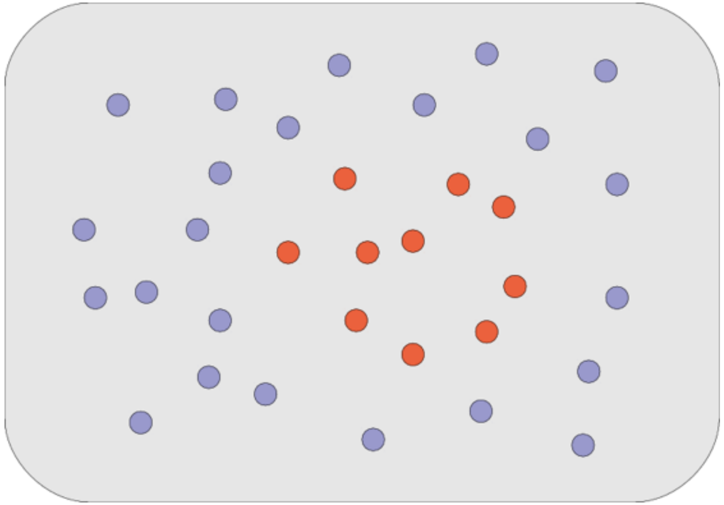
- Lagrangian **dual** problem is:

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{j,k} \alpha_j \alpha_k y_j y_k \left(x_j^T x_k\right)$$

subject to $\forall i$: $\alpha_i \geq 0$, and $0 = \sum_{i=1}^{N} \alpha_i y_i$

6 Kernelization Tricks in SVMs

Non-linear Case



- Not all classes can be separated via a hyperplane
- Essential:
  - Dual representation uses only the product of data instances:

$$\hat{f}(x) = \sum_{i=1}^{N} \alpha_i \cdot y_i \cdot \boxed{x_i^T x} + b$$

- x_i : i-th training instance
- α_i : weight for i-th training instance

- Same for the Lagrangian...

Feature engineering using $\phi(x)$

Cf. lecture on regression. Chapter "beyond linear input"

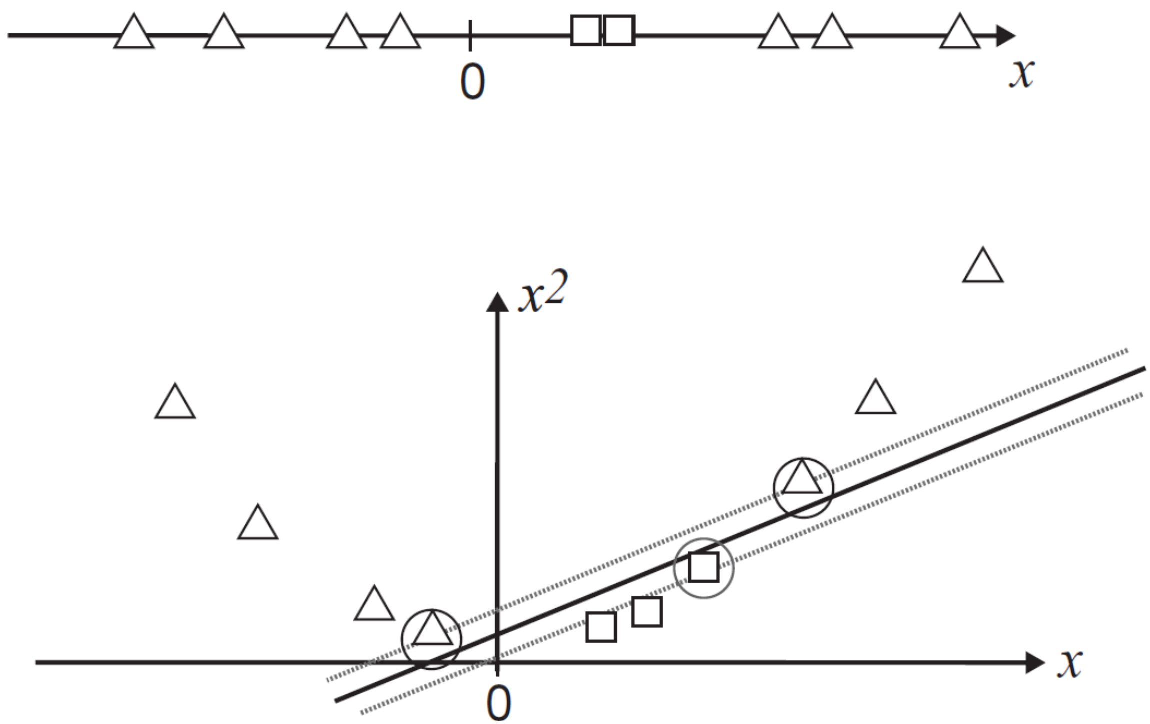
- Classifier: Given $x_i \in \mathbb{R}^d$, $\phi: \mathbb{R}^d \to \mathbb{R}^D, w \in \mathbb{R}^D$

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

- Learning:

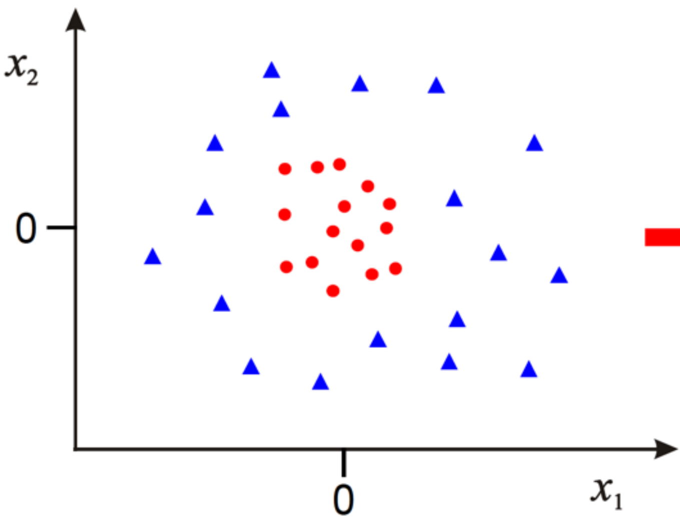
$$\min_{w \in \mathbb{R}^D} \frac{1}{C} \|w\|^2 + \sum_{i=1}^{N} \max(0, 1 - y_i \hat{f}(x_i))$$

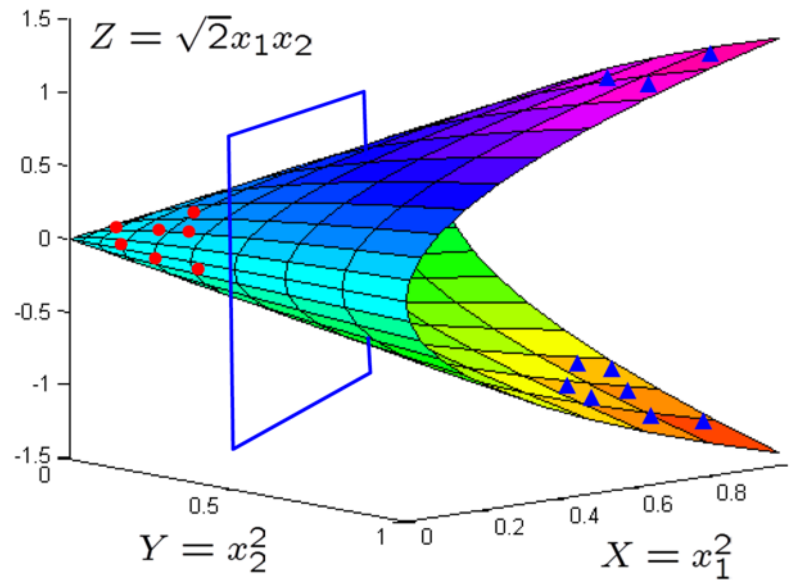
Example 1: From 1-dim to 2-dim



Example 2: From 2-dim to 3-dim

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \to \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} \qquad \mathbb{R}^2 \to \mathbb{R}^3$$





- Data is linearly separable in 3D

(Zisserman 2015)

- This means that the problem can still be solved by a linear classifier

Feature engineering using $\phi(x)$

- Classifier: Given $x_i \in \mathbb{R}^d$, $\phi: \mathbb{R}^d \to \mathbb{R}^D, w \in \mathbb{R}^D$

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

- Learning:

$$\min_{w \in \mathbb{R}^D} \frac{1}{C} \|w\|^2 + \sum_{i=1}^{N} \max(0, 1 - y_i \hat{f}(x_i))$$

- $\phi(x)$ maps to high dimensional space $\mathbb{R}^D$ where data is separable
- If $D \gg d$ then there are many more parameters to learn for w

Dual classifier in transformed feature space

Classifier:

$$\hat{f}(x) = \sum_{i=1}^{N} \alpha_i \cdot y_i \cdot x_i^T x \; + b$$

$$\Longrightarrow \; \hat{f}(x) = \sum_{i=1}^{N} \alpha_i \cdot y_i \cdot \phi(x_i)^T \phi(x) + b$$

Learning:

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{j,k} \alpha_j \alpha_k y_j y_k \left(x_j^T x_k\right)$$

$$\Longrightarrow \max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{j,k} \alpha_j \alpha_k y_j y_k \left(\phi\left(x_j\right)^T \phi(x_k)\right)$$

subject to $\forall i$: $\alpha_i \geq 0$, and $0 = \sum_{i=1}^{N} \alpha_i y_i$

ϕ(x)
only occurs in pairs
$\phi\left(x_j\right)^T \phi(x_i)$

Kernels
$k\left(x_j, x_i\right) = \phi\left(x_j\right)^T \phi(x_i)$

Dual classifier using kernels

Classifier:

$$\hat{f}(x) = \sum_{i=1}^{N} \alpha_i \cdot y_i \cdot \phi(x_i)^T \phi(x) + b$$

$$\Longrightarrow \hat{f}(x) = \sum_{i=1}^{N} \alpha_i \cdot y_i \cdot k(x_i, x) + b$$

Learning:

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{j,k} \alpha_j \alpha_k y_j y_k \left(\phi(x_j)^T \phi(x_k) \right)$$

$$\Longrightarrow \max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{j,k} \alpha_j \alpha_k y_j y_k k(x_j, x_k)$$

subject to $\forall i$: $\alpha_i \geq 0$, and $0 = \sum_{i=1}^{N} \alpha_i y_i$

Example kernels

- Linear kernels: $k(x, x') = x^T x'$

- Polynomial kernels: $k(x, x') = (1 + x^T x')^d, \text{for any } d > 0$
  - Contains all polynomial terms up to degree

- Gaussian kernels: $k(x, x') = e^{-\frac{\|x - x'\|^2}{2\sigma^2}}$, for $\sigma > 0$
  - Infinite dimensional feature space
  - Also called Radial basis function kernel (RBF)
    - often works quite well!

- Graph kernels: random walk
- String kernels: ...
- **build your own kernel for your own problem!**

Summary on kernels

- „Instead of inventing funny non-linear features, we may directly invent funny kernels" (Toussaint 2019)
- Inventing a kernel is intuitive:
  - $k(x, x')$ expresses how correlated y and y' should be
  - it is a meassure of similarity, it compares x and x'.
- Specifying how 'comparable' x and x' are is often more intuitive than defining "features that might work".

Background reading and more

- Smooth readong about SVMs: Alexandre Kowalczyk, Support vector machines succinctly. Syncfusion. Free ebook: https://www.syncfusion.com/ebooks/support_vector_machines_succinctly
  - **Also talks about most efficient algorithms to be used for finding support vectors (it is neither of the two presented here!)**

7 Transductive Classification

Transductive learning characteristics

Characteristics

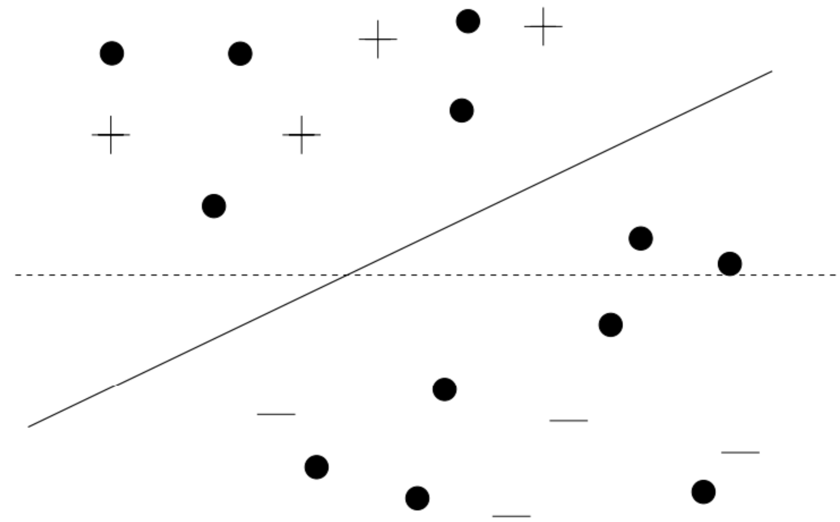
- Training data AND test data known at learning time
- Learning happens specifically for the given test cases

Use cases

- news recommender
- spam classifier
- document reorganization

Thorsten Joachims:
Transductive Inference for Text Classification using Support Vector Machines. *ICML 1999: 200-209*

Maximum margin hyperplane



Training data $\{\ldots, (\vec{x}_i, y_i), \ldots\}$

Test data $\{\ldots \vec{x}_j^* \ldots\}$

Loss function

$$\frac{1}{2} \parallel \vec{w} \parallel^2 + C \sum_{i=0}^{n} \xi_i + C^* \sum_{j=0}^{k} \xi_j^*$$

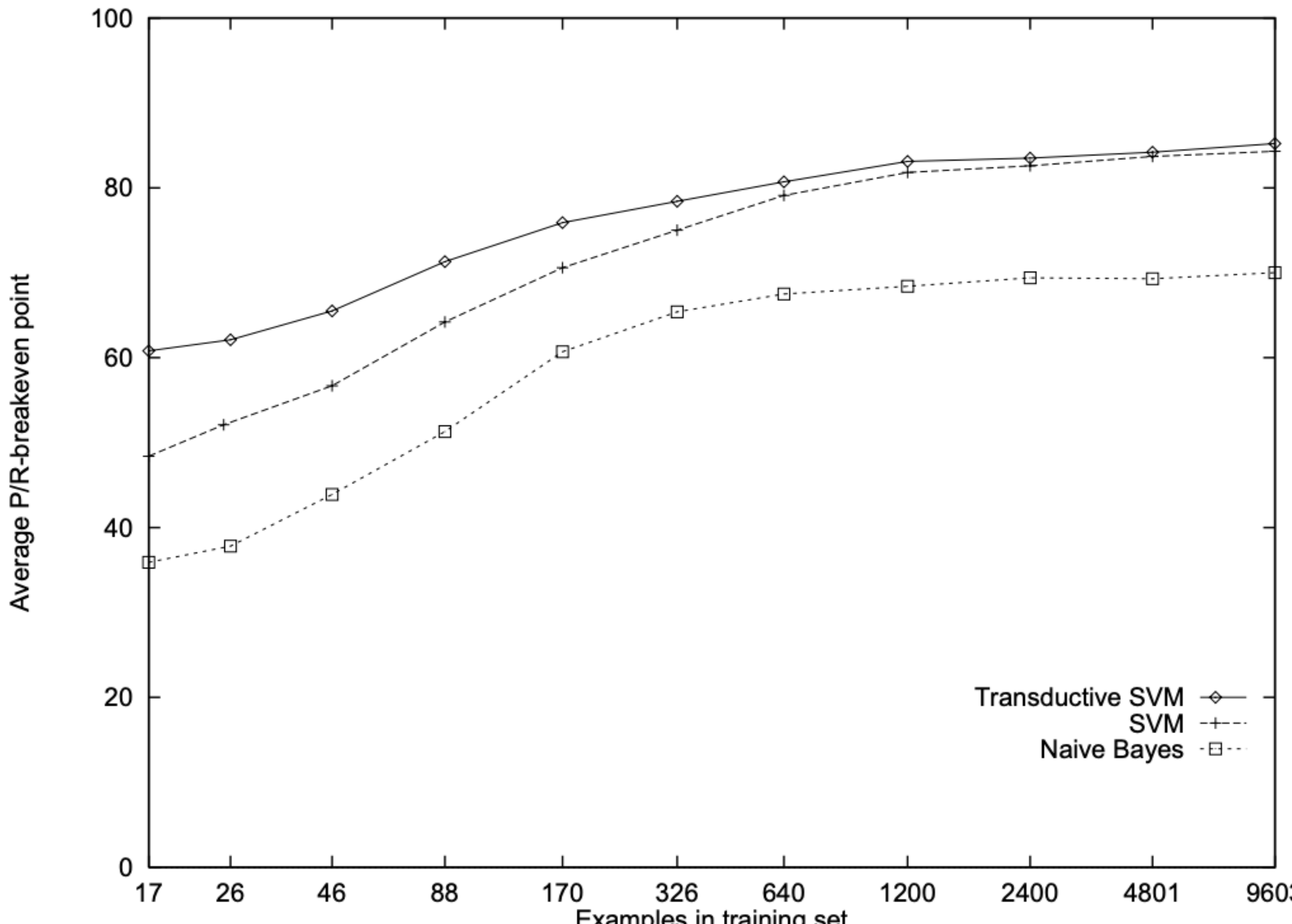
subject to:

$$\forall_{i=1}^{n}: y_i[\vec{w} \cdot \vec{x}_i + b] \geq 1 - \xi_i$$
$$\forall_{j=1}^{k}: y_j^*[\vec{w} \cdot \vec{x}_j^* + b] \geq 1 - \xi_j^*$$
$$\forall_{i=1}^{n}: \xi_i > 0$$
$$\forall_{j=1}^{k}: \xi_j^* > 0$$

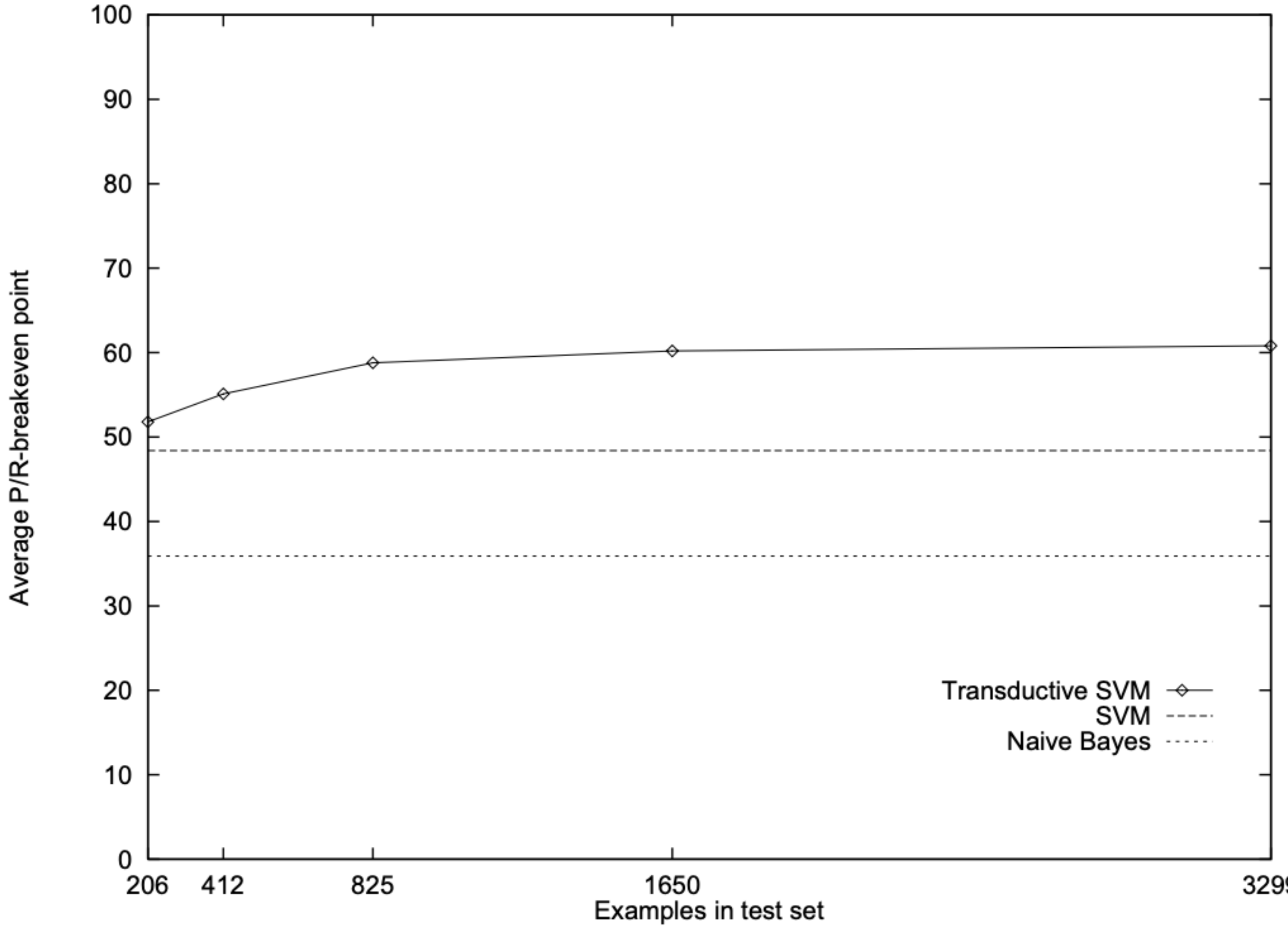
Naive, intractable approach:

- for every hyperplane:
  - classify $\vec{x}_j^*$
  - compute loss

Reuters data set experiments (3299 test documents)



Reuters data set experiments (17 training documents)



Current research at Analytic Computing:

Multi-label classification with hyperbolic hyperplanes

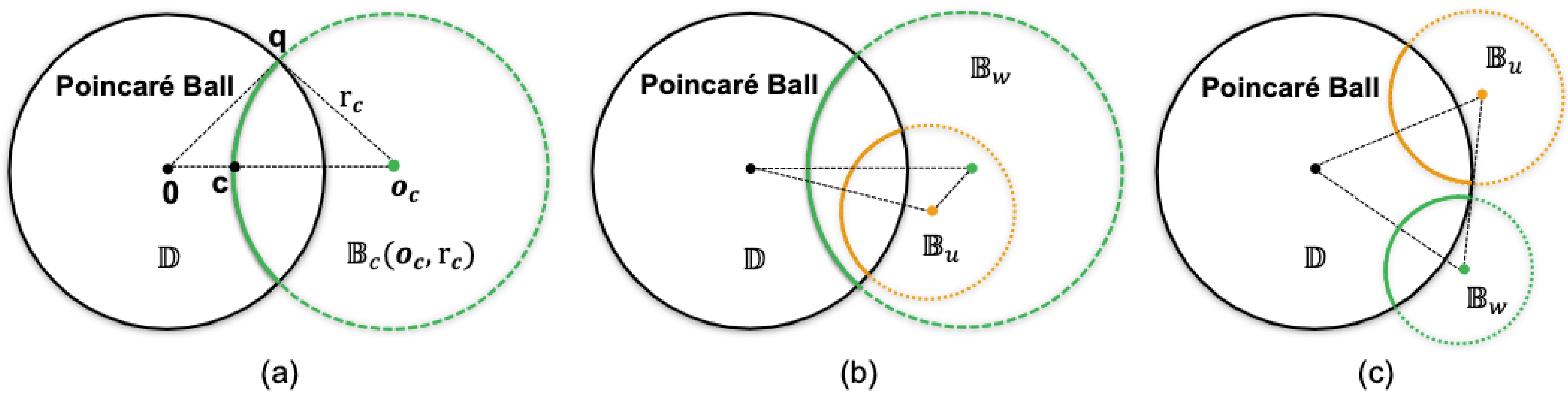


Figure 2: (a) A Poincaré hyperplane is defined as the intersection between the Poincaré ball $\mathbb{D}$ and the boundary of an n-ball $\mathbb{B}_\mathbf{c}$. The Poincaré hyperplane is uniquely parameterized by a center point $\mathbf{c}$, and the corresponding n-ball (its radius and center) can be uniquely determined by Proposition 1. (b) Label implication is interpreted as n-ball insideness. (c) Mutual exclusion is interpreted as n-ball disjointedness.

So far with hyperbolic logistic regression but
doing it with hyperbolic SVM could be a project or a bachelor thesis

Universität Stuttgart
KI

Thank you!

Steffen Staab

E-Mail Steffen.staab@ki.uni-stuttgart.de
Telefon +49 (0) 711 685-88100
www.ki.uni-stuttgart.de/

Universität Stuttgart
Analytic Computing, KI
Universitätsstraße 32, 50569 Stuttgart