# Support Vector Machines II

Alexander Schönhuth

UNIVERSITÄT BIELEFELD
Faculty of Technology

Bielefeld University
July 6, 2023

# Learning Goals Today / Overview

- Supervised learning: summary
- Support vector machines

*Supervised Learning*

# Supervised Learning

- There is a functional relationship

$$f^* : \mathbb{R}^d \to V$$

we would like to understand, or *learn*.
  - *Regression*: $V = \mathbb{R}$
  - *Classification*: $V = \{1, ..., k\}$
- To learn it, we are given *m data points*

$$(x_i, f^*(x_i) = y_i)_{i=1,...,m}$$

that reflect this functional relationship.

*Final goal*: Predict $f^*(x)$ well on unknown data points $x$.

# Supervised versus Unsupervised Learning

- *Unsupervised Learning:*
  - Given unlabeled data

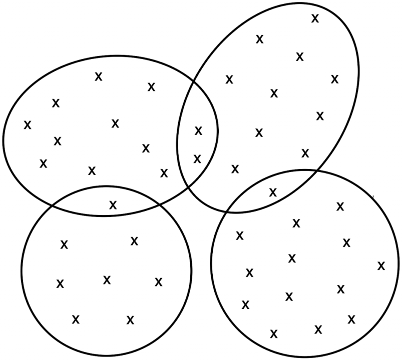
$$(x_i)_{i=1,\ldots,m}$$

  - *Goal:* Infer subgroups of data points
  - *Alternative Problem Formulation:* Learn the probability distribution

$$\mathbf{P}(\mathbf{X})$$

    that governs the generation of data points

# Unsuperviseed Learning: Example

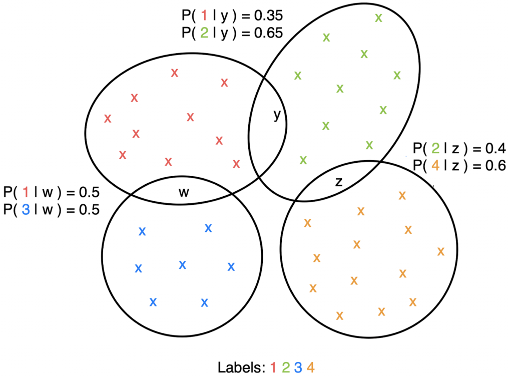



Generative distribution yielding four clusters

# Supervised versus Unsupervised Learning

- *Supervised Learning:*
  - Given labeled data
  $$(x_i, y_i)_{i=1,\dots,m}$$
  - *Goal:* Learn functional relationship $f^* : \mathbb{R}^d \to V$, s.t. $y_i = f^*(x_i)$
  - *Alternative Problem Formulation:* Learn the probability distribution
  $$\mathbf{P}(\mathbf{X}, \mathbf{y}) \quad \text{or} \quad \mathbf{P}(\mathbf{y} \mid \mathbf{X})$$
  as a more general version of functional relationship

# Unsuperviseed Learning: Example




Generative distribution yielding four clusters *and* corresponding labels

## SUPERVISED LEARNING: TRAINING

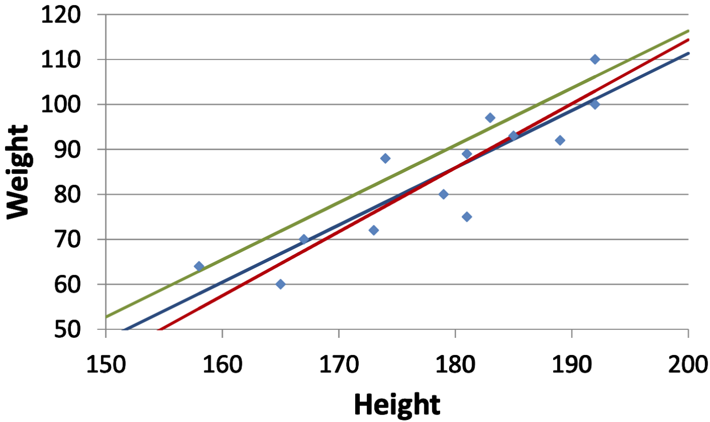
- The idea is to set up a *training procedure* (an algorithm) that *learns* $f^*$ from the training data.
- Learning $f^*$ means to *approximate* it by $f : \mathbb{R}^d \to V$ sufficiently well, where $f \in \mathcal{M}$ for a certain class of functions $\mathcal{M}$.
- In most cases, $f \in \mathcal{M}$ are parameterized by parameters $\mathbf{w}$. This means that we have to pick an appropriate choice of parameters $\mathbf{w}$ for learning $f^*$.

# Supervised Learning

- We need to determine a *cost (or loss) function* $C$ where $C(f, f^*)$ measures how well $f \in \mathcal{M}$ approximates $f^*$.
- *Optimization*: Pick $f \in \mathcal{M}$ (by picking the right set of parameters) that yields small (possibly minimal) cost $C(f, f^*)$
- *Generalization*: Optimization procedure should address that $f$ is to approximate $f^*$ well on *unknown data points*.

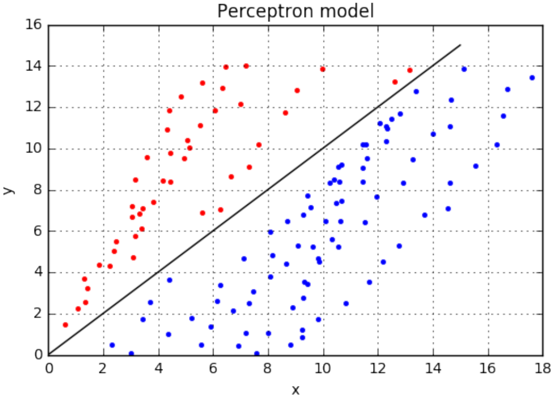
# LINEAR REGRESSION

EXAMPLE: $f : \mathbb{R} \to \mathbb{R}$



# PERCEPTRON

EXAMPLE: $f : \mathbb{R}^2 \to \{0, 1\}$




$$\begin{aligned} f \quad \mathbb{R}^2 &\longrightarrow \{0 = \text{blue}, 1 = \text{red}\} \\ (x_1, x_2) &\mapsto \begin{cases} 1 & x_2 - x_1 > 0 \\ 0 & x_2 - x_1 \leq 0 \end{cases} \end{aligned} \tag{1}$$

UNIVERSITÄT BIELEFELD

We need to specify:

- How to set up the data being used for training
- A model class $\mathcal{M}$, for example linear functions
- A cost function $C(f, f^*)$ that evaluates the goodness of $f \in \mathcal{M}$
- An optimization procedure that picks $f$ such that $C(f, f^*)$ is minimal, or very small
- Keep in mind that $f$ is to perform well on previously unseen data

# Supervised Learning

## Notation

- The dataset is given by a *design matrix* $\mathbf{X} \in \mathbb{R}^{m \times d}$ where $m$ is the number of data points and $d$ is the number of *features*
- Each data point $x_i$ (a row in $\mathbf{X}$) is assigned to a *label* $y_i$ that reflects the true functional relationship $y_i = f^*(x_i)$, where further $\mathbf{y} = (y_1, ..., y_m) \in V^m$ is the *label vector*.

*Generalization*

# Enabling Generalization: Data

## Training, Test and Validation

- Split $(\mathbf{X}, \mathbf{y})$ into
  - training data $(\mathbf{X}^{(\text{train})}, \mathbf{y}^{(\text{train})})$
  - validation data $(\mathbf{X}^{(\text{val})}, \mathbf{y}^{(\text{val})})$
  - test data $(\mathbf{X}^{(\text{test})}, \mathbf{y}^{(\text{test})})$
- *Training data:*
  - Used to pick the optimal set of parameters
  - That is, pick the optimal, particular element of $\mathcal{M}$
  - Training reflects common optimization procedure

# Enabling Generalization: Data

## Training, Test and Validation

- Split $(\mathbf{X}, \mathbf{y})$ into
  - training data $(\mathbf{X}^{(\text{train})}, \mathbf{y}^{(\text{train})})$
  - validation data $(\mathbf{X}^{(\text{val})}, \mathbf{y}^{(\text{val})})$
  - test data $(\mathbf{X}^{(\text{test})}, \mathbf{y}^{(\text{test})})$
- *Validation data:*
  - Used to determine *hyperparameters*
  - Hyperparameters refer to number of training iterations, choosing optimization procecure, neural network architecture variants
  - Some reflect selecting appropriate subsets of $\mathcal{M}$

# Enabling Generalization: Data

## Training, Test and Validation

- Split $(\mathbf{X}, \mathbf{y})$ into
  - training data $(\mathbf{X}^{(\text{train})}, \mathbf{y}^{(\text{train})})$
  - validation data $(\mathbf{X}^{(\text{val})}, \mathbf{y}^{(\text{val})})$
  - test data $(\mathbf{X}^{(\text{test})}, \mathbf{y}^{(\text{test})})$
- *Nested training cycle:*
  1. Train on training data using current hyperparameters
     ☞ Yields parameters
  2. Evaluate determined parameters on validation data
     ☞ Adjusting hyperparameters yields new hyperparameters
  3. Return to 1.
- Nested training yields optimal parameters and hyperparameters

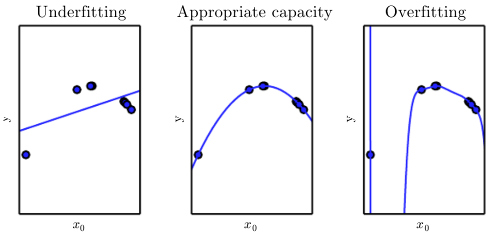
# Enabling Generalization: Data

## Training, Test and Validation

- Split $(\mathbf{X}, \mathbf{y})$ into
  - training data $(\mathbf{X}^{(\text{train})}, \mathbf{y}^{(\text{train})})$
  - validation data $(\mathbf{X}^{(\text{val})}, \mathbf{y}^{(\text{val})})$
  - test data $(\mathbf{X}^{(\text{test})}, \mathbf{y}^{(\text{test})})$
- *Test data:*
  - $(\mathbf{X}^{(\text{test})}, \mathbf{y}^{(\text{test})})$ are never touched during training
  - Final goal is to minimize cost on test data
- *Machine learning dilemma:* Optimize with respect to data you do not know

# Enabling Generalization: Model

## Capacity, Under- and Overfitting




Left: Linear functions underfit
Center: Polynomials of degree 2 neither under- nor overfit
Right: Polynomials of degree 9 overfit

- Choose a class of models that has the right *capacity*
- Capacity too large: *overfitting*
- Capacity too small: *underfitting*

# Enabling Generalization: Cost Function

## Regularization

Let $C(f, f^*)$ be the cost function. Let $\mathbf{w} = (w_1, ..., w_k)$ be the parameters specifying elements of $f_\mathbf{w} \in \mathcal{M}$.

- Usually, $C$ refers to only known data points. That is, $C$ evaluates as
$$C(f, f^*) = \sum_i C(f(x_i), y_i = f^*(x_i)) \tag{2}$$
where $x_i$ runs over all training data points.
- Add a *regularization term* to cost function, and choose $f_\mathbf{w}$ that yields minimal
$$C(f_\mathbf{w}, f^*) + \lambda\Omega(\mathbf{w}) \tag{3}$$
- $\lambda$ is a hyperparameter

# Enabling Generalization: Cost Function

## Regularization

- Prominent examples:
  - *$L_1$ norm*: $\Omega(\mathbf{w}) := \sum_i |w_i|$
  - *$L_2$ norm*: $\Omega(\mathbf{w}) := \sum_i w_i^2$
- Rationale: Penalize too many non-zero weights
- Virtually less complex model, hence virtually less capacity
- ☞ Prevents overfitting, yields better generalization

Optimization can be an iterative procedure.

- *Early stopping*: Stop the optimization procedure before cost function reaches an optimum on the training data.
- *Dropout*: Randomly fix parameters to zero, and optimize remaining parameters.

# Enabling Generalization: Summary

- Training reflects an optimization procedure
- *Insight:* Optima correspond to overfitting training data
- *Solution:* Seek to output parameters “nearby” optima
- Nearly all generalization techniques address this:
  - Early stopping stops optimization before optimum is reached
  - Dropout carries out optimization in pre-set lower-dimensional subspace
  - Regularization forces to watch out for optima in lower-dimensional subspaces

*Prominent Supervised Learning Model Examples*

# Linear Regression

- Design matrix $\mathbf{X} \in \mathbb{R}^{m \times d}$, label vector $\mathbf{y} \in \mathbb{R}^m$
- Model class: Let $\mathbf{w} \in \mathbb{R}^d$

$$f_{\mathbf{w}} = f(\mathbf{x}; \mathbf{w}) : \begin{array}{ccc} \mathbb{R}^d & \longrightarrow & \mathbb{R} \\ \mathbf{x} & \mapsto & \mathbf{w}^T\mathbf{x} \end{array} \tag{4}$$

- *Remark*: Note that the case $\mathbf{w}^T\mathbf{x} + b$ can be treated as a special case to be included in $\mathcal{M}$, by augmenting vectors $\mathbf{x}_i$ by an entry 1 (think about this...)
- Cost function (recall $y_i = f^*(\mathbf{x}_i)$)

$$C(f, f^*) := \frac{1}{m}||(f(\mathbf{x}_1), ..., f(\mathbf{x}_m)) - \mathbf{y}||_2^2 = \frac{1}{m}\sum_{i=1}^{m}(f(\mathbf{x}_i) - \mathbf{y}_i)^2 \tag{5}$$

# Linear Regression

## Optimization

- Solve for

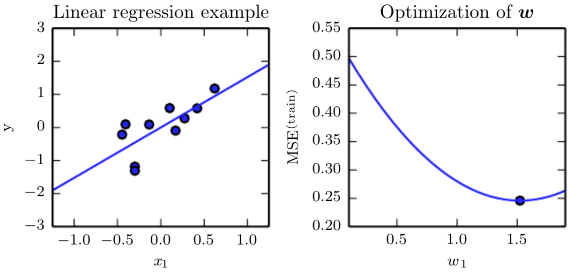
$$\nabla_{\mathbf{w}} C(f_{\mathbf{w}}, f^*) = 0 \tag{6}$$

to achieve a minimum. This yields the *normal equations*

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{7}$$

- *Global optimum* if $\mathbf{X}^T\mathbf{X}$ is invertible
- Do this on *training data* (so $\mathbf{X} = \mathbf{X}^{(\text{train})}, \mathbf{y} = \mathbf{y}^{(\text{train})}$) only. Hope that cost on test data is small.

# NORMAL EQUATIONS




- *Left*: Data points, and the linear function $y = w_1 x$ that approximates them best
- *Right*: Mean squared error (MSE) depending on $w_1$
- *Remark on Perceptrons*: Optimizing is different, but also supported by a very easy optimization scheme (the *perceptron algorithm*)

# Nearest Neighbor Classification

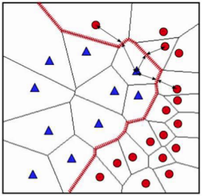
- Consider appropriate distance measure

$$D : \mathbb{R}^d \times \mathbb{R}^d \longrightarrow \mathbb{R}_+ \tag{8}$$

- For unknown data point $\mathbf{x}$, determine the closest given data point

$$\mathbf{x}_{i^*} := \text{argmin}_i(D(\mathbf{x}, \mathbf{x}_i)) \tag{9}$$

- Predict label of $\mathbf{x}$ as $y_{i^*}$



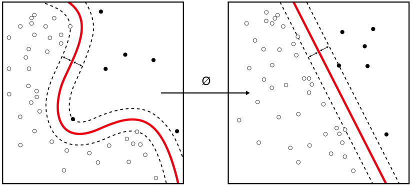
# Support Vector Machines

- *Realization*: From (7), write

$$\mathbf{w}^T\mathbf{x} = \sum_{i=1}^{m} \alpha_i \mathbf{x}^T\mathbf{x}_i = \sum_{i=1}^{m} \alpha_i \langle \mathbf{x}, \mathbf{x}_i \rangle \tag{10}$$

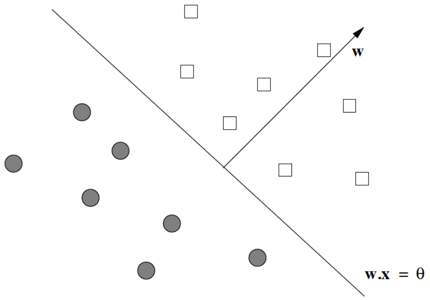
- Replace $\langle ., . \rangle$ by different *kernel* (i.e. scalar product) $k(., .)$, that is by computing $\langle \phi(.), \phi(.) \rangle$ for appropriate $\phi$

☞ Seek $\alpha$'s to maximize margin: still easy to optimize both for regression and classification!



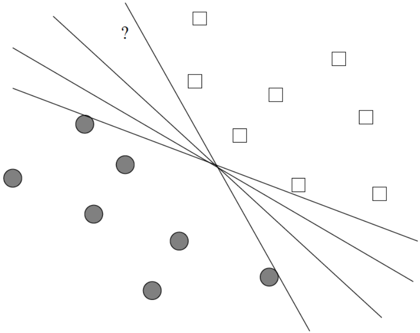
*Support Vector Machines*

# Perceptron Revisited




- A perceptron divides the space into two half spaces
- Half spaces capture the two different classes
- Normal vector alternative description of half space

# Perceptron Revisited

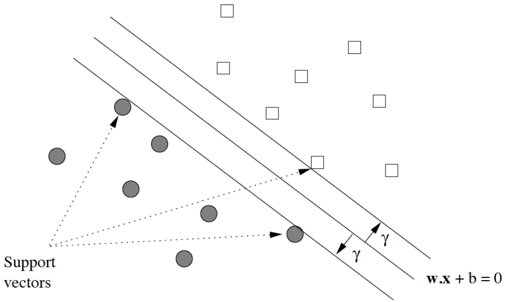



- Several half spaces (normal vectors) divide training data
- *Question:* any half space optimal, in a sensibly defined way?
- What to do if data cannot be separated (is *non-separable*)?

# Support Vector Machines: Motivation

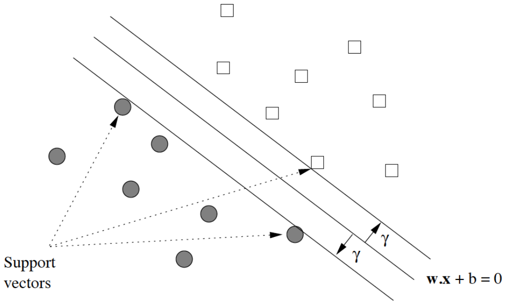
- Support vector machines (SVM's) address to choose most reasonable half space
- SVM's choose half space that maximizes the *margin*
- If separable, maximize distance between hyperplane and closest data points
- If not separable, minimize *loss function* that
  - penalizes misclassified points
  - penalizes points correctly classified by too close to hyperplane (to a lesser extent)

# SEPARABLE DATA




- *Goal:* Select hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$ that maximizes distance $\gamma$
- *Intuition:* The further away data from hyperplane, the more certain their classification
- Increases chances to correctly classify unseen data (to generalize)

# Support Vectors




- Two parallel hyperplanes at distance $\gamma$ touch one or more of *support vectors*
- In most cases, $d$-dimensional data set has $d + 1$ support vectors (but there can be more)

# Problem Formulation: First Try

Let $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)$ be a training data set, where $\mathbf{x}_i \in \mathbb{R}^d, y_i \in \{-1, +1\}, i = 1, ..., n$.

Problem: By varying $\mathbf{w}, b$, maximize $\gamma$ such that

$$y_i(\mathbf{w}\mathbf{x}_i + b) \geq \gamma \quad \text{for all } i = 1, ..., n \tag{11}$$

## Issue

- Replacing $\mathbf{w}$ and $b$ by $2\mathbf{w}$ and $2b$ yields $y_i(2\mathbf{w}\mathbf{x}_i + 2b) \geq 2\gamma$
- There is no optimal $\gamma$

Problem badly formulated ☞ try harder!

# Problem Formulation: Solution

- Data set $(\mathbf{x}_i, y_i), i = 1, ..., n$ as before
- *Solution:* Impose additional constraint: consider only combinations $\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$ such that for support vectors $\mathbf{x}$

$$y_i(\mathbf{w}\mathbf{x} + b) \in \{-1, +1\} \tag{12}$$

- *Good Formulation:* By varying $\mathbf{w}, b$, maximize $\gamma$ such that

$$y_i(\mathbf{w}\mathbf{x}_i + b) \geq \gamma \quad \text{for all } i = 1, ..., n \tag{13}$$

  and (12) applies

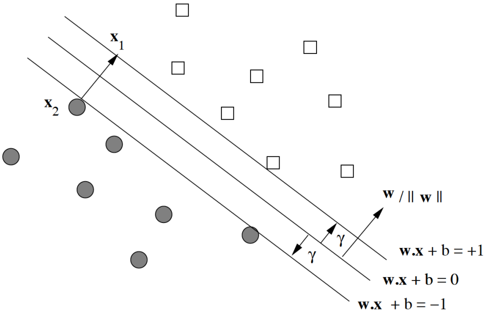
# Problem Formulation: Observation

- Let
$$d(\mathbf{x}_i, H) := \min_x \{d(\mathbf{x}_i, \mathbf{x}) \mid \mathbf{w}\mathbf{x} + b = 0\} \tag{14}$$
be the distance between $\mathbf{x}_i$ and $H$.
- Basic linear algebra implies that
$$d(\mathbf{x}_i, H) = \frac{1}{||\mathbf{w}||} |\mathbf{w}\mathbf{x}_i + b| \tag{15}$$

☞ (13) maximizes distance between hyperplane and data points

☞ (12) ensures that support vectors are at distance $1/||\mathbf{w}||$

# Alternative Problem Formulation I




- $\mathbf{w}, b, \gamma$ determined according to (12),(13)
- $\mathbf{x}_2$ is support vector on lower hyperplane, so by (12), $\mathbf{w}\mathbf{x}_2 + b = -1$
- Let $\mathbf{x}_1$ be the projection of $\mathbf{x}_2$ onto upper hyperplane:

$$\mathbf{x}_1 = \mathbf{x}_2 + 2\gamma\frac{\mathbf{w}}{||\mathbf{w}||} \qquad (16)$$

# Alternative Problem Formulation II

That is, further, $\mathbf{x}_1$ is on the hyperplane defined by $\mathbf{wx} + b = 1$, meaning

$$\mathbf{wx}_1 + b = 1 \tag{17}$$

Substituting (16) into (17) yields

$$\mathbf{w} \cdot (\mathbf{x}_2 + 2\gamma \frac{\mathbf{w}}{||\mathbf{w}||}) + b = 1 \tag{18}$$

By further regrouping, we obtain

$$\mathbf{wx}_2 + b + 2\gamma \frac{\mathbf{ww}}{||\mathbf{w}||} = 1 \tag{19}$$

Because $\mathbf{ww} = ||\mathbf{w}||^2$, by further regrouping, we conclude that

$$\gamma = \frac{1}{||\mathbf{w}||} \tag{20}$$

# Alternative Problem Formulation III

Let dataset $(\mathbf{x}_i, y_i), i = 1, ..., n$ be as before.

Equivalent Problem Formulation:

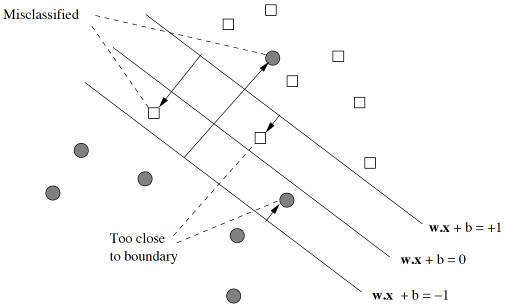
By varying $\mathbf{w}, b$, minimize $||\mathbf{w}||$ subject to

$$y_i(\mathbf{w}\mathbf{x}_i + b) \geq 1 \quad \text{for all } i = 1, ..., n \tag{21}$$

Optimizing under Constraints

- Topic is broadly covered
- Many packages can be used
- Target function $\sum_i w_i^2$ quadratic; well manageable

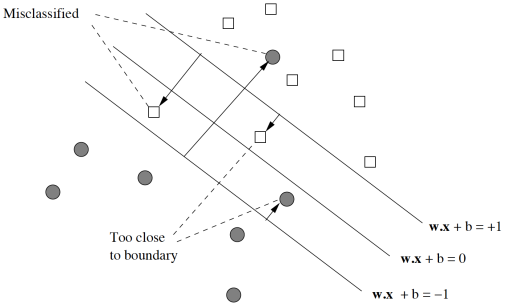
# Non Separable Data Sets




Situation:

- Some points misclassified, some too close to boundary
  ☞ *bad points*
- *Non separable data:* any choice of **w**, $b$ yields bad points

# NON SEPARABLE DATA: MOTIVATION




- *Situation:* No hyperplane can separate the data points correctly
- *Approach:*
  - Determine appropriate penalties for bad points
  - Solve original problem, by involving penalties

# Non Separable Data: Motivation II

Let $(\mathbf{x}_i, y_i), i = 1, ...n$ be training data, where

- $\mathbf{x}_i = (x_{i1}, ..., x_{id})$,
- $y_i \in \{-1, +1\}$

and let $\mathbf{w} = (w_1, ..., w_d)$.

*Minimize* the following function:

$$f(\mathbf{w}, b) = \frac{1}{2} \sum_{j=1}^{d} w_j^2 + C \sum_{i=1}^{n} \max\{0, 1 - y_i(\sum_{j=1}^{d} w_j x_{ij} + b)\} \qquad (22)$$

# Non Separable Data: Motivation II

$$f(\mathbf{w}, b) = \underbrace{\frac{1}{2}\sum_{j=1}^{d} w_j^2}_{\text{Seek minimal } ||\mathbf{w}||} + \underbrace{C\sum_{i=1}^{n} \max\{0, 1 - y_i(\sum_{j=1}^{d} w_j x_{ij} + b)\}}_{\text{Bad point penalty}}$$

- Minimizing $||\mathbf{w}||$ equivalent to minimizing monotone function of $||\mathbf{w}||$
  ☞ Minimizing $f$ seeks minimal $||\mathbf{w}||$
- Vectors $\mathbf{w}$ and training data balanced in terms of basic units:

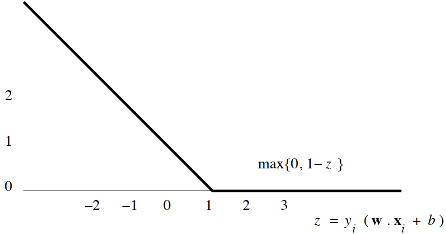
$$\frac{\partial(||\mathbf{w}||^2/2)}{\partial w_i} = w_i \quad \text{and} \quad \frac{\partial(\sum_{j=1}^{d} w_j x_{ij} + b)}{\partial w_i} = x_{ij}$$

- $C$ is a regularization parameter
  - Large $C$: minimize misclassified points, but accept narrow margin
  - Small $C$: accept misclassified points, but widen margin

# Non Separable Data: Hinge Function

Let the *hinge function* $L$ be defined by

$$L(\mathbf{x}_i, y_i) = \max\{0, 1 - y_i(\sum_{j=1}^{d} w_j x_{ij} + b)\} \tag{23}$$




- $L(\mathbf{x}_i, y_i) = 0$ iff $\mathbf{x}_i$ on the correct side of hyperplane with sufficient margin
- The worse $\mathbf{x}_i$ is located the greater $L(\mathbf{x}_i, y_i)$

# Non Separable Data: Hinge Function

Let the *hinge function* $L$ be defined by

$$L(\mathbf{x}_i, y_i) = \max\{0, 1 - y_i(\sum_{j=1}^{d} w_j x_{ij} + b)\}$$

Partial derivatives of hinge function:

$$\frac{\partial L}{\partial w_j} = \begin{cases} 0 & \text{if } y_i(\sum_{j=1}^{d} w_j x_{ij} + b) \geq 1 \\ -y_i x_{ij} & \text{otherwise} \end{cases} \tag{24}$$

Reflecting:

- If $\mathbf{x}_i$ is on right side with suffcient margin: nothing to be done
- Otherwise adjust $w_j$ to have $\mathbf{x}_i$ better placed

# General / Further Reading

Literature

- Mining Massive Datasets, Sections 12.1–12.3
  http://infolab.stanford.edu/~ullman/mmds/ch10.pdf