Support Vector Machines

Lecture series „Machine Learning“

Niels Landwehr

Research Group „Data Science“
Institute of Computer Science
University of Hildesheim

Agenda for Lecture

- SVMs for linearly separable data
- SVMs for not linearly separable data
- SVMs and kernels

Review: Supervised Learning for Binary Classification

- **Review: Supervised learning for binary classification**

- Instance space $\mathcal{X}$, for this lecture assuming $\mathcal{X} = \mathbb{R}^M$
- Target space $\mathcal{Y}$, for this lecture assuming two classes that are encoded as $\mathcal{Y} = \{-1, 1\}$. Note that the encoding of classes is different from what we had earlier, as this makes the following mathematics more convenient

- Training data $\mathcal{D} = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$

- Want to learn a model that can classify instances $\mathbf{x} \in \mathcal{X}$. For this lecture, we define the model to have the form

$$f_{\boldsymbol{\theta}} : \mathbb{R}^M \rightarrow \mathbb{R}$$

and make a classification decision by

$$\hat{y} = \begin{cases} 1 : f_{\boldsymbol{\theta}}(\mathbf{x}) \geq 0 \\ -1 : f_{\boldsymbol{\theta}}(\mathbf{x}) < 0 \end{cases}$$

Review: Linear Model

- Assume that $f_{\boldsymbol{\theta}}$ is given by a linear model, that is,

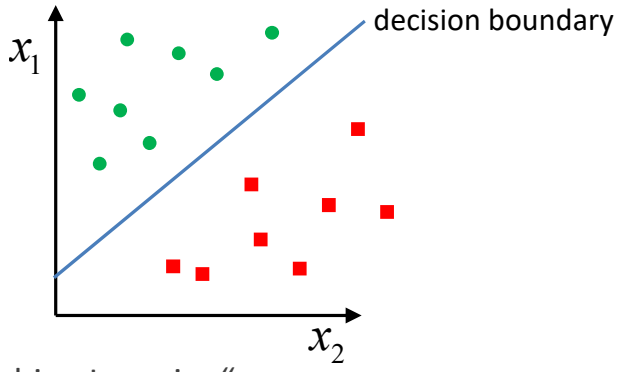
$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\boldsymbol{\theta} + b \qquad \boldsymbol{\theta} \in \mathbb{R}^M, b \in \mathbb{R}$$

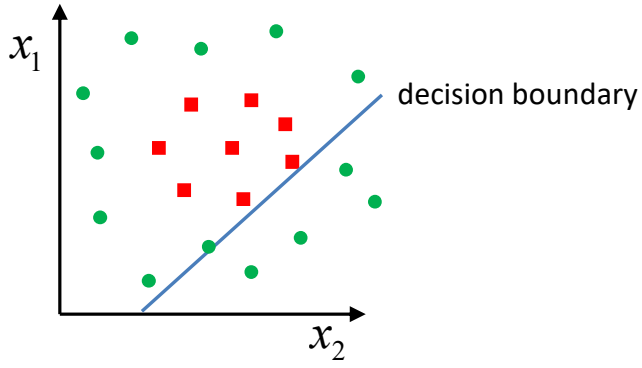
- As for the logistic regression model, the decision boundary of the model will be linear
  - The decision boundary is the set of points $\mathbf{x} \in \mathbb{R}^M$ for which

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\boldsymbol{\theta} + b \ = 0$$

  - This set of points is a line for $\mathcal{X} = \mathbb{R}^2$, and a hyperplane for general $\mathcal{X} = \mathbb{R}^M$

- Data can be linearly separable (that is, there is a linear model with error zero) or not:





Linear Separability

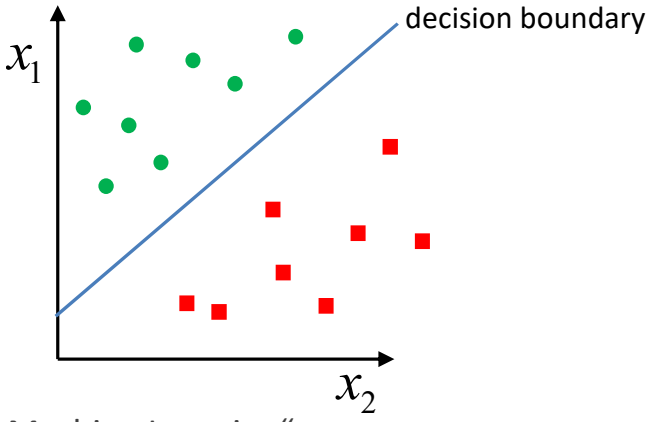
- More formally, a linear model is said to **separate** data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ if and only if

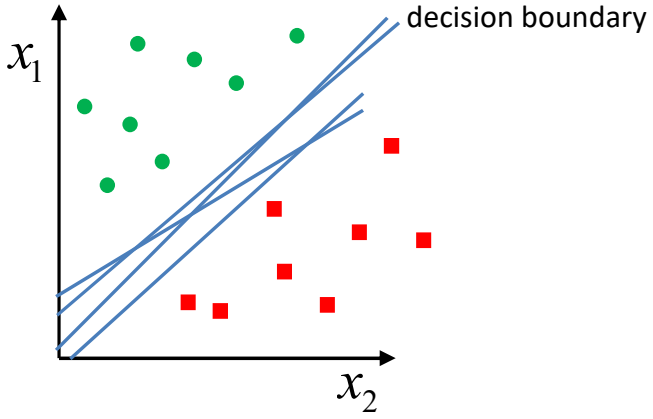
$$f_{\boldsymbol{\theta}}(\mathbf{x}_n) > 0 \qquad \text{for all } \mathbf{x}_n \text{ with } y_n = +1$$
$$f_{\boldsymbol{\theta}}(\mathbf{x}_n) < 0 \qquad \text{for all } \mathbf{x}_n \text{ with } y_n = -1$$

or, equivalenty,

$$f_{\boldsymbol{\theta}}(\mathbf{x}_n) y_n > 0 \qquad \text{for all } \mathbf{x}_n \in \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$$

- If there is a separating linear model, there are typically many:





Prelude: The Perceptron Classifier

- Learning the linear model: based on training data $\mathcal{D} = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$, find a suitable weight vector $\boldsymbol{\theta}$ and bias b
- We have already seen a solution to this problem in the lecture on linear classification, in the form of the logistic regression model
- In this lecture, we will look at a different method for learning linear classification models, called support vector machines (or SVMs for short)

- As an introductory example, let's first look at a simple learning algorithm called the **perceptron algorithm**:
  - Initialize $\boldsymbol{\theta}^{(0)} = \mathbf{0},\ b^{(0)} = 0$
  - Loop through training data points $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)$ several times and update for each *n*

$$\boldsymbol{\theta}^{(t+1)} := \boldsymbol{\theta}^{(t)} + \eta y_n \mathbf{x}_n$$
$$b^{(t+1)} := b^{(t)} + \eta y_n$$

where the hyperparameter $\eta > 0$ is a learning rate

Prelude: The Perceptron Classifier

- If the training data is linearly separable and the learning rate η is sufficiently small, the perceptron algorithm will converge to a separating linear model
- Intuition: why does the perceptron algorithm work?
  - If $y_n = 1$ it holds that $f_{\boldsymbol{\theta}^{(t+1)}}(\mathbf{x}_n) > f_{\boldsymbol{\theta}^{(t)}}(\mathbf{x}_n)$:

$$\begin{aligned} f_{\boldsymbol{\theta}^{(t+1)}}(\mathbf{x}_n) &= \mathbf{x}_n^{\mathrm{T}}(\boldsymbol{\theta}^{(t)} + \eta\mathbf{x}_n) + b^{(t)} + \eta \\ &= \mathbf{x}_n^{\mathrm{T}}\boldsymbol{\theta}^{(t)} + b^{(t)} + \eta\mathbf{x}_n^{\mathrm{T}}\mathbf{x}_n + \eta \\ &= f_{\boldsymbol{\theta}^{(t)}}(\mathbf{x}_n) + \eta\mathbf{x}_n^{\mathrm{T}}\mathbf{x}_n + \eta \end{aligned}$$

According to update rule on last slide, plugging in $y_n = 1$

$\eta\mathbf{x}^{\mathrm{T}}\mathbf{x} + \eta = \eta \| x \|_2^2 + \eta > 0$

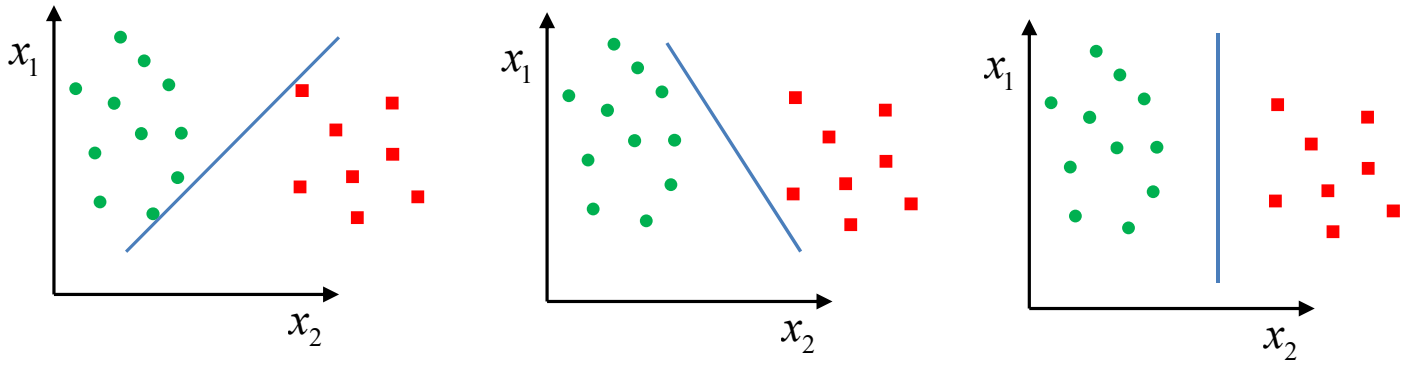
  - If $y_n = -1$ it holds that $f_{\boldsymbol{\theta}^{(t+1)}}(\mathbf{x}_n) < f_{\boldsymbol{\theta}^{(t)}}(\mathbf{x}_n)$:

$$\begin{aligned} f_{\boldsymbol{\theta}^{(t+1)}}(\mathbf{x}_n) &= \mathbf{x}_n^{\mathrm{T}}(\boldsymbol{\theta}^{(t)} - \eta\mathbf{x}_n) + b^{(t)} - \eta \\ &= \mathbf{x}_n^{\mathrm{T}}\boldsymbol{\theta}^{(t)} + b^{(t)} - \eta\mathbf{x}_n^{\mathrm{T}}\mathbf{x}_n - \eta \\ &= f_{\boldsymbol{\theta}^{(t)}}(\mathbf{x}_n) - \eta\mathbf{x}_n^{\mathrm{T}}\mathbf{x}_n - \eta \end{aligned}$$

- Therefore, if there is a wrong prediction ($y_n = +1$ and $f_{\boldsymbol{\theta}^{(t+1)}}(\mathbf{x}_n) < 0$ or $y_n = -1$ and $f_{\boldsymbol{\theta}^{(t+1)}}(\mathbf{x}_n) > 0$) the prediction is corrected through the parameter updates

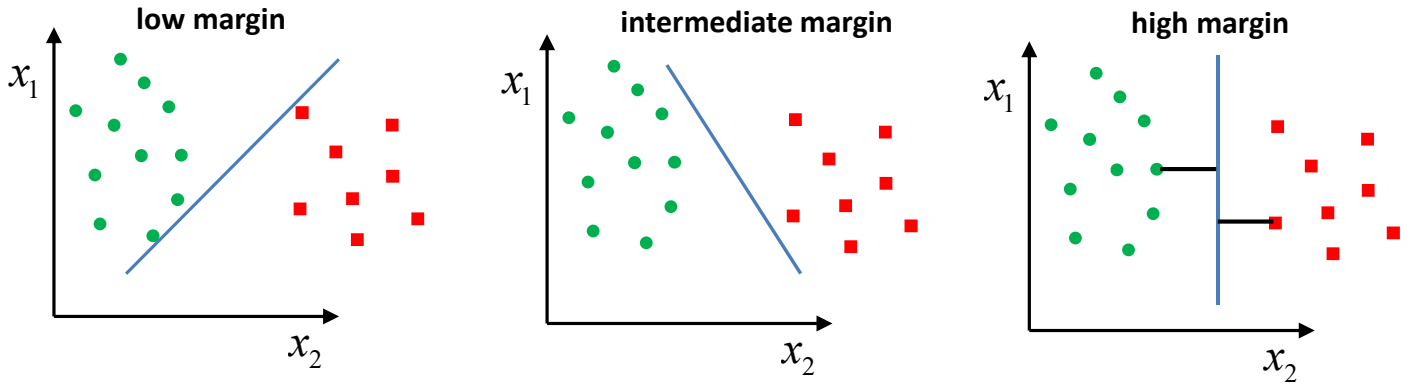
Which Separating Linear Model?

- **For the discussion within this section, we assume given data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ that is linearly separable (will relax that assumption later in the lecture)**
- Which is the best separating linear model (hyperplane) out of all possible ones?

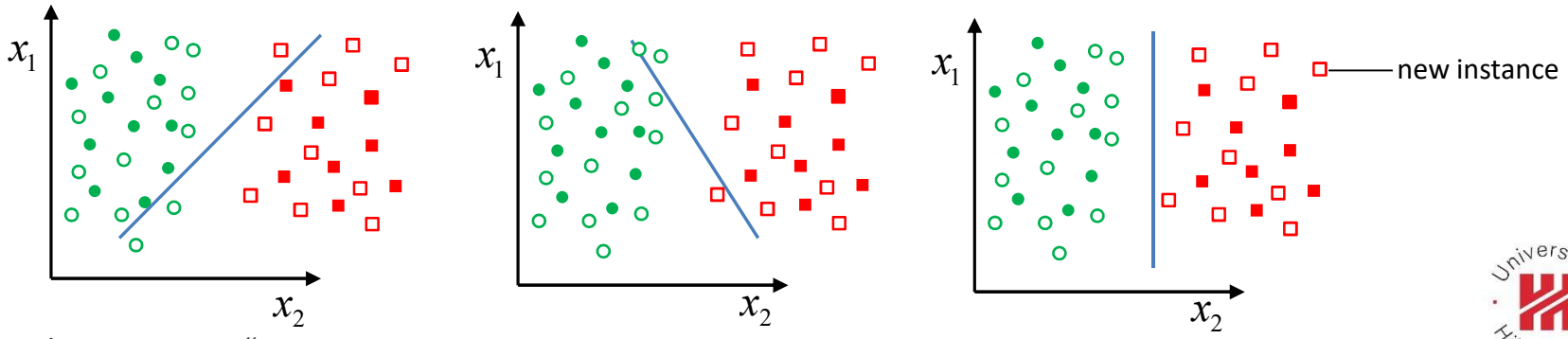


Which Separating Linear Model?

- **For the discussion within this section, we assume given data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ that is linearly separable (will relax that assumption later in the lecture)**
- Which is the best separating linear model (hyperplane) out of all possible ones?



- Intuition: the one with highest **margin** (highest distance of a training instance to the hyperplane) is best, because it most likely gives correct predictions on new instances

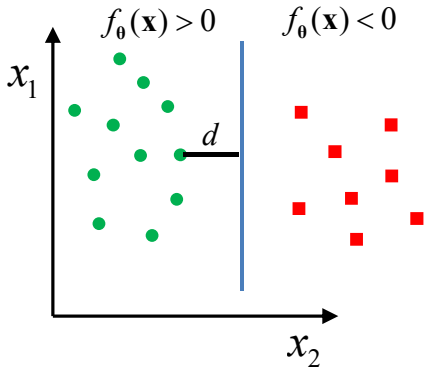


Maximum Margin Criterion

- Without proof: The signed distance of a point $\mathbf{x} \in \mathbb{R}^M$ to the hyperplane given by the linear model $\mathbf{x}^T\boldsymbol{\theta} + b$ can be computed as

$$d = \frac{\mathbf{x}^T\boldsymbol{\theta} + b}{\|\boldsymbol{\theta}\|} = \frac{f_{\boldsymbol{\theta}}(\mathbf{x})}{\|\boldsymbol{\theta}\|}$$

$$d > 0 \text{ if } f_{\boldsymbol{\theta}}(\mathbf{x}) > 0$$
$$d < 0 \text{ if } f_{\boldsymbol{\theta}}(\mathbf{x}) < 0$$



- To find the maximum-margin hyperplane, we can solve the following optimization problem:

$$\text{Find } C, \boldsymbol{\theta}, b$$
$$\text{such that } y_n \frac{\mathbf{x}_n^T\boldsymbol{\theta} + b}{\|\boldsymbol{\theta}\|} \geq C \quad \text{for } n \in \{1, \ldots, N\}$$
$$\text{and } C \text{ maximal}$$

For positive instances, enforce a minimum signed distance of +*C*, for negative instances, a signed distance of -*C*

Normalizing Normal Vector of Hyperplane

- Support vector machines learn a linear model with maximum margin from data

- First of all we note that for a linear model $f_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\boldsymbol{\theta} + b$ with predictions

$$\hat{y} = \begin{cases} 1 : f_{\boldsymbol{\theta}}(\mathbf{x}) \geq 0 \\ -1 : f_{\boldsymbol{\theta}}(\mathbf{x}) < 0 \end{cases}$$

  we can rescale the parameters without changing the predictions: the model $\mathbf{x}^{\mathrm{T}}(\lambda\boldsymbol{\theta}) + (\lambda b)$ gives the same predictions as the model $\mathbf{x}^{\mathrm{T}}\boldsymbol{\theta} + b$

- For the optimization problem from last slide, for any potential solution $C, \boldsymbol{\theta}, b$, we can normalize $\boldsymbol{\theta}, b$ such that $\|\boldsymbol{\theta}\| = 1/C$ and obtain an equivalent solution that satisfies

$$y_n \frac{\mathbf{x}_n{}^{\mathrm{T}}\boldsymbol{\theta} + b}{1/C} \geq C \qquad \text{for all } n \in \{1, \ldots, N\}$$

$$\Leftrightarrow y_n(\mathbf{x}_n{}^{\mathrm{T}}\boldsymbol{\theta} + b) \geq 1 \qquad \text{for all } n \in \{1, \ldots, N\}$$

Optimization Criterion

- After normalization, the minimal distance of a training point to the hyperplane is $C = 1 / \|\boldsymbol{\theta}\|$ (simply because $\|\boldsymbol{\theta}\| = 1 / C$)

- We can therefore reformulate the optimization problem for the maximum-margin hyperplane as

$$\begin{aligned} &\text{Find } \boldsymbol{\theta}, b \\ &\text{such that } y_n(\mathbf{x}_n{}^{\mathrm{T}}\boldsymbol{\theta} + b) \geq 1 \quad \text{for } n \in \{1,...,N\} \\ &\text{and } \|\boldsymbol{\theta}\| \text{ minimal} \end{aligned}$$

  or equivalently (and easier to optimize):

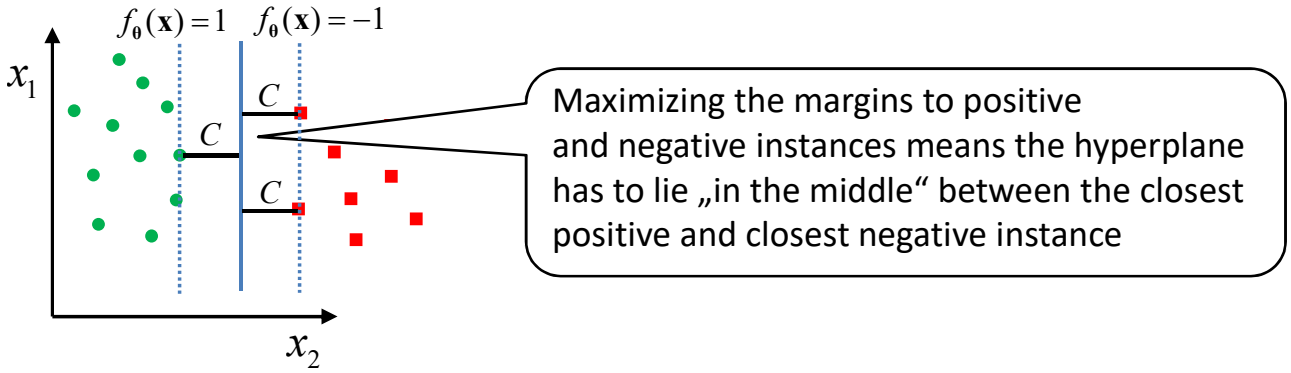
$$\begin{aligned} &\text{Find } \boldsymbol{\theta}, b \\ &\text{such that } y_n(\mathbf{x}_n{}^{\mathrm{T}}\boldsymbol{\theta} + b) \geq 1 \quad \text{for } n \in \{1,...,N\} \\ &\text{and } \frac{1}{2}\|\boldsymbol{\theta}\|^2 \text{ minimal} \end{aligned}$$

- This is called the **primal hard-margin** SVM optimization problem, and the resulting classifier a **primal hard-margin SVM**

Terminology: Margin and Support Vectors

- This is a quadratic minimization problem with linear inequality constraints
- Such problems have a unique minimum

- A note on notation/terminology:
  - It is clear that for a solution of the optimization problem, the distance of the closest positive instance to the hyperplane will be *C* and the distance of the closest negative instance to the hyperplane will also be *C*



  - These closests instances are called **support vectors** (typically multiple instances)
  - The quantity $2C$ (or equivalently $2/||\boldsymbol{\theta}||$) is called the **margin**

Dual Problem

- Alternatively, we can formulate a so-called dual version of the hard-margin SVM
- In the dual problem we find a dual parameter vector $\boldsymbol{\alpha} \in \mathbb{R}^N$ by solving

$$\boldsymbol{\alpha}^* = \arg\max_{\boldsymbol{\alpha}} \tilde{L}(\boldsymbol{\alpha})$$

$$\text{subject to } \sum_{n=1}^{N} \alpha_n y_n = 0 \text{ and } \alpha_n \geq 0 \text{ for all n} \in \{1,...,N\}$$

$$\text{where } \tilde{L}(\boldsymbol{\alpha}) = -\frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m \mathbf{x}_n^{\mathrm{T}} \mathbf{x}_m + \sum_{n=1}^{N} \alpha_n$$

- For the solution $\boldsymbol{\alpha}^* \in \mathbb{R}^N$ it holds that

$$\boldsymbol{\theta}^* = \sum_{n=1}^{N} \alpha_n^* y_n \mathbf{x}_n$$

where $\boldsymbol{\theta}^*$ is the solution of the primal hard-margin SVM problem given above

Dual Problem

- The dual problem (last slide) is again a quadratic optimization problem, but with N rather than M optimization variables and different (simpler) constraints than the primal problem

- Depending on the number of features and number of data points, either the primal or the dual formulation can be easier (faster) to solve

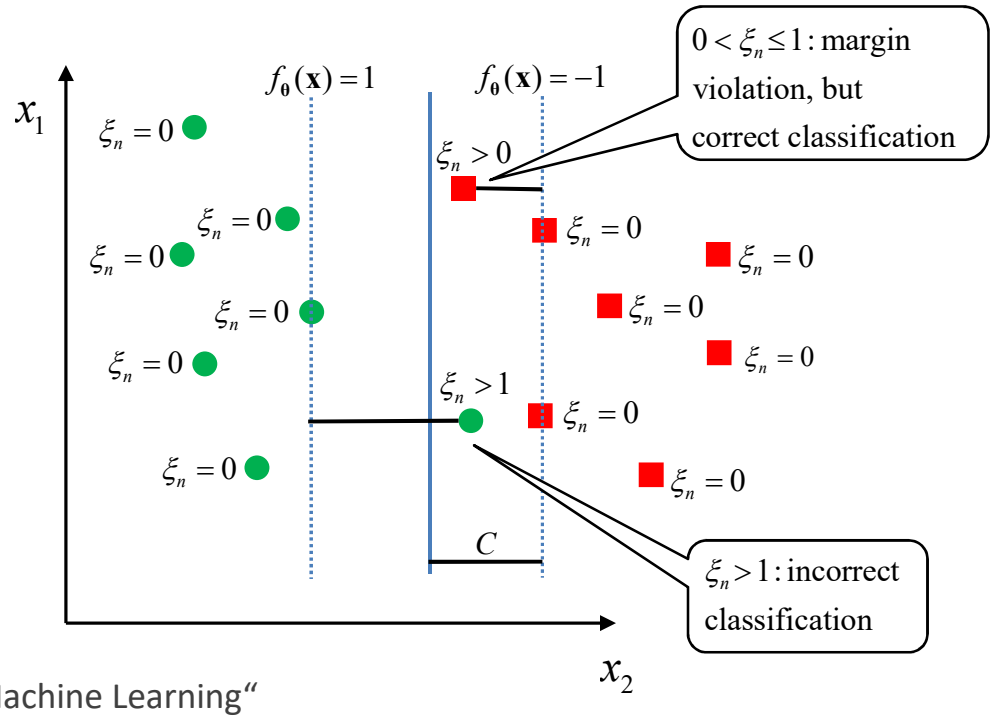
Agenda for Lecture

- SVMs for linearly separable data
- SVMs for not linearly separable data
- SVMs and kernels

The Non-Separable Case: Slack Variables

- The approach discussed so far only works for linearly separable data: if the data is not linearly separable, no hyperplane with positive margin exists
- It is also called the **hard-margin** solution, as it strictly enforces that all examples have a positive margin (lie at the right side of the hyperplane with a certain distance)

- **Idea**: introduce so-called „slack variables“ ξ_n to allow that some instances violate the hard margin constraints:



$$\xi_n = \max(0, \underbrace{1 - y_n f_\theta(\mathbf{x}_n)}_{\text{negative if instance } \mathbf{x}_n \text{ is on correct side of margin}})$$

The Non-Separable Case: Optimization Problem

- With slack variables, the goal is to maximize the margin while at the same time minimizing the margin violations as measured by the slack variables:

maximize margin

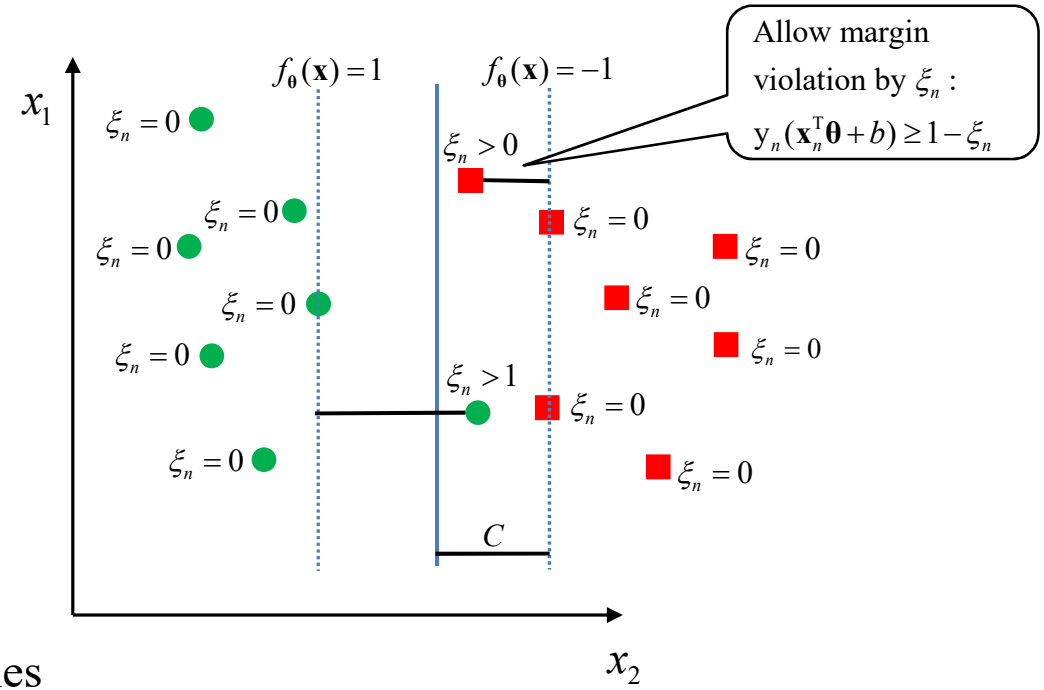
minimize margin violations

$$\text{Find } \boldsymbol{\theta}, b, \boldsymbol{\xi}$$

$$\text{such that } \frac{1}{2}\|\boldsymbol{\theta}\|^2 + \gamma \sum_{n=1}^{N} \xi_n \text{ minimal}$$

$$\text{with } y_n(\mathbf{x}_n{}^{\mathrm{T}}\boldsymbol{\theta} + b) \geq 1 - \xi_n \text{ and } \xi_n \geq 0$$

constraint enforces that margin is violated no more than ξ_n



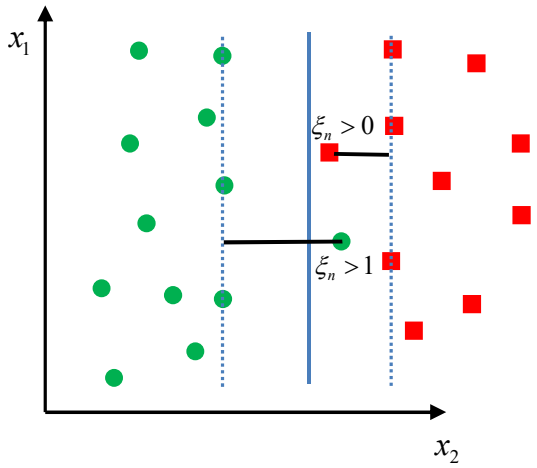
$\boldsymbol{\xi} = (\xi_1, \ldots, \xi_N)$ vector of all slack variables

$\gamma \in \mathbb{R}$ trade-off hyperparameter

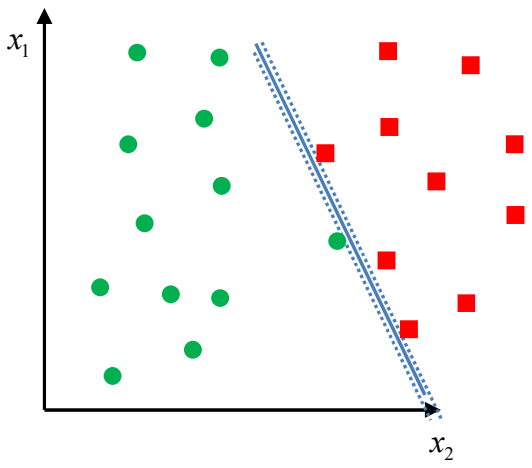
The Non-Separable Case: Optimization Problem

- The solution of this optimization problem is called the (primal) **soft-margin** solution, compared to the **hard-margin** solution without the slack variables given above
- The linear classifier obtained by solving it is also called the (primal) soft margin support vector machine (or SVM)
- Even if the data is linearly separable, it is often preferable to use the soft-margin solution, as it can be more robust to outliers

Soft margin solution



Hard margin solution



- Using parameter γ, can smoothly trade-off between slack and margin ($\gamma \to \infty$ corresponds to hard-margin solution)

Substituting for Slack Variables: Hinge Loss

- It is clear that for any solution of the problem

$$\text{Find } \boldsymbol{\theta}, b, \boldsymbol{\xi}$$
$$\text{such that } \frac{1}{2}\|\boldsymbol{\theta}\|^2 + \gamma \sum_{n=1}^{N} \xi_n \text{ minimal}$$
$$\text{with } y_n(\mathbf{x}_n{}^{\mathrm{T}}\boldsymbol{\theta} + b) \geq 1 - \xi_n \text{ and } \xi_n \geq 0$$

it has to hold that $\xi_n = \max(0, 1 - y_n f_{\boldsymbol{\theta}}(\mathbf{x}_n))$:

The constraint $y_n f_{\boldsymbol{\theta}}(\mathbf{x}_n) \geq 1 - \xi_n$ implies $\xi_n \geq 1 - y_n f_{\boldsymbol{\theta}}(\mathbf{x}_n)$, and with $\xi_n \geq 0$ it follows that $\xi_n \geq \max(0, 1 - y_n f_{\boldsymbol{\theta}}(\mathbf{x}_n))$. Given that we minimize over ξ_n, $\xi_n = \max(0, 1 - y_n f_{\boldsymbol{\theta}}(\mathbf{x}_n))$ has to hold (and this also ensures that all constraints are fulfilled)

- We can therefore reformulate the optimization by plugging in $\xi_n = \max(0, 1 - y_n f_{\boldsymbol{\theta}}(\mathbf{x}_n))$:

$$\arg\min_{\boldsymbol{\theta}, b} \frac{1}{2}\|\boldsymbol{\theta}\|^2 + \gamma \sum_{n=1}^{N} \max(0, 1 - y_n f_{\boldsymbol{\theta}}(\mathbf{x}_n))$$

Substituting for Slack Variables: Hinge Loss

- This is simply the minimization of a regularized loss function:

$$\arg\min_{\boldsymbol{\theta},b} \frac{1}{2}\|\boldsymbol{\theta}\|^2 + \gamma \sum_{n=1}^{N} \max(0, 1 - y_n f_{\boldsymbol{\theta}}(\mathbf{x}_n))$$

$$= \arg\min_{\boldsymbol{\theta},b} \frac{1}{N} \sum_{n=1}^{N} \ell(f_{\boldsymbol{\theta}}(\mathbf{x}_n), y_n) + \underbrace{\frac{1}{2N\gamma}}_{\text{regularization weight}} \|\boldsymbol{\theta}\|^2$$

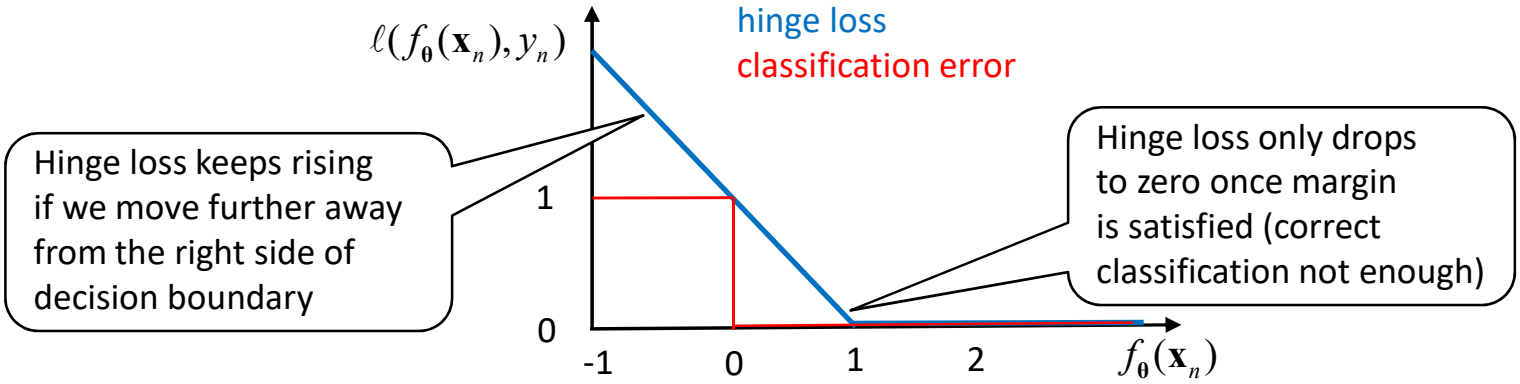
where $\ell(f_{\boldsymbol{\theta}}(\mathbf{x}_n), y_n) = \max(0, 1 - y_n f_{\boldsymbol{\theta}}(\mathbf{x}_n))$

- The loss function $\ell(f_{\boldsymbol{\theta}}(\mathbf{x}_n), y_n)$ is called the **hinge loss**
- Minimizing the combination of hinge loss and L2-regularizer above will again give us the solution to the (primal, soft-margin) SVM

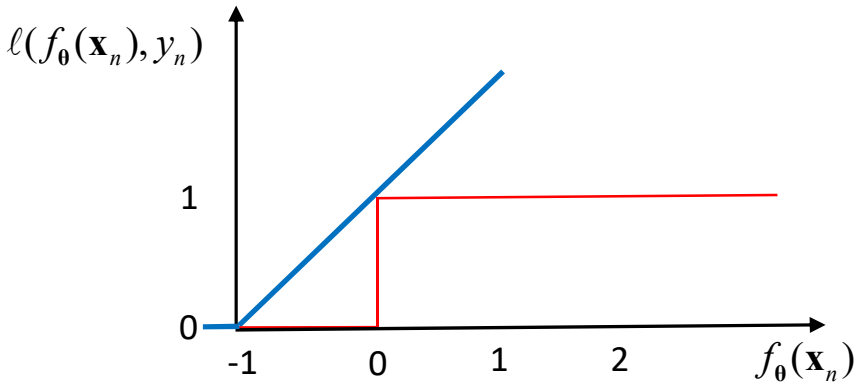
- We can say that the „geometric“ approach to learning we have taken (by looking at the margin) has led us to a new loss function

Substituting for Slack Variables: Hinge Loss

- Plotting the hinge loss function
  - If $y_n = 1$, the loss is $\max(0, 1 - f_{\boldsymbol{\theta}}(\mathbf{x}_n)) = \begin{cases} 0 : f_{\boldsymbol{\theta}}(\mathbf{x}_n) > 1 \\ 1 - f_{\boldsymbol{\theta}}(\mathbf{x}_n) : \text{otherwise} \end{cases}$



  - If $y_n = -1$:



Substituting for Slack Variables: Hinge Loss

- The primal optimization problem

$$\arg\min_{\boldsymbol{\theta},b} \frac{1}{N}\sum_{n=1}^{N}\max(0,1-y_n f_{\boldsymbol{\theta}}(\mathbf{x}_n)) + \frac{1}{2N\gamma}\|\boldsymbol{\theta}\|^2$$

  can be solved by gradient descent

- The gradient of the loss on a single training instance is given by

$$\frac{\partial}{\partial \boldsymbol{\theta}}\max(0,1-y_n f_{\boldsymbol{\theta}}(\mathbf{x}_n)) = \begin{cases} 0 & : y_n f_{\boldsymbol{\theta}}(\mathbf{x}_n) \geq 1 \\ -y_n \mathbf{x}_n & : y_n f_{\boldsymbol{\theta}}(\mathbf{x}_n) < 1 \end{cases}$$

$$\frac{\partial}{\partial b}\max(0,1-y_n f_{\boldsymbol{\theta}}(\mathbf{x}_n)) = \begin{cases} 0 & : y_n f_{\boldsymbol{\theta}}(\mathbf{x}_n) \geq 1 \\ -y_n & : y_n f_{\boldsymbol{\theta}}(\mathbf{x}_n) < 1 \end{cases}$$

- The gradient of the regularizer is $\frac{\partial}{\partial \boldsymbol{\theta}}\frac{1}{2N\gamma}\|\boldsymbol{\theta}\|^2 = \frac{1}{N\gamma}\boldsymbol{\theta}$

- Objective is convex, global minimum will be found
- Problem can be solved on very large data sets using stochastic gradient descent

Dual Optimization for Soft Margin Solution

- Similarly as for the hard-margin case, the optimization problem

$$\text{Find } \boldsymbol{\theta}, b, \boldsymbol{\xi} \text{ such that } \frac{1}{2}\|\boldsymbol{\theta}\|^2 + \gamma \sum_{n=1}^{N} \xi_n \text{ minimal}$$

$$\text{with } y_n(\mathbf{x}_n{}^{\mathrm{T}}\boldsymbol{\theta} + b) \geq 1 - \xi_n \text{ and } \xi_n \geq 0$$

has a dual equivalent. It is given by

$$\boldsymbol{\alpha} = \arg\max_{\boldsymbol{\alpha}} \tilde{L}(\boldsymbol{\alpha})$$

$$\text{subject to } \sum_{n=1}^{N} \alpha_n y_n = 0, \quad \alpha_n \leq \gamma, \quad \alpha_n \geq 0$$

$$\text{where } \tilde{L}(\boldsymbol{\alpha}) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m \mathbf{x}_n^{\mathrm{T}} \mathbf{x}_m + \sum_{n=1}^{N} \alpha_n$$

- Again, it holds that

$$\boldsymbol{\theta}^* = \sum_{n=1}^{N} \alpha_n^* y_n \mathbf{x}_n$$

Solving the Dual Optimization Problems

- The dual optimization problems (both the hard-margin and the soft-margin case) require the minimization of a quadratic function with linear equality and inequality constraints

- These problems can be solved with well-established optimization algorithms (no details here)

- Generally, the difference to the primal formulation is that $|\boldsymbol{\alpha}| = N$ while $|\boldsymbol{\theta}| = M$
  - Using the dual formulation has advantages if M is large and N is not too large
  - For very large N, use primal solution with stochastic gradient descent (can also deal with quite large M if needed)

- Dual solutions have another important advantage, namely that **kernels** can be easily employed: remaining section of lecture

Agenda for Lecture

- SVMs for linearly separable data
- SVMs for not linearly separable data
- SVMs and kernels

Dual Prediction

- Solving the dual SVM problem gives us a vector $\boldsymbol{\alpha}^* \in \mathbb{R}^N$
- Recap: at the solution $\boldsymbol{\alpha}^*$, it holds that

$$\boldsymbol{\theta}^* = \sum_{n=1}^{N} \alpha_n^* y_n \mathbf{x}_n$$

- From this it follows that we can have a dual prediction by

$$f_{\boldsymbol{\theta}^*}(\mathbf{x}) = \mathbf{x}^{\mathrm{T}} \boldsymbol{\theta}^* + b$$

$$= \sum_{n=1}^{N} \alpha_n^* y_n \mathbf{x}^{\mathrm{T}} \mathbf{x}_n + b$$

- In the dual solution, it holds that $\alpha_n^* = 0$ for all points that satisfy the margin constraint, that is, that are not support vectors (no proof). If most points do, this leads to significant computational savings in the dual prediction

- The dual solution does not give us b, however, b can be recovered by (without proof)

$$b = \frac{1}{|N_{SV}|} \sum_{n \in N_{SV}} \left(y_n - \sum_{m=1}^{N} \alpha_m^* y_m \mathbf{x}_m^{\mathrm{T}} \mathbf{x}_n \right) \qquad N_{SV} = \{n \in \{1,\ldots,N\} \mid \alpha_n^* > 0\}$$

-

Dual Solutions and Feature Maps

- In the dual case, as stated above we solve the optimization problem

$$\boldsymbol{\alpha} = \arg\max_{\boldsymbol{\alpha}} \tilde{L}(\boldsymbol{\alpha})$$

$$\text{subject to } \sum_{n=1}^{N} \alpha_n y_n = 0, \quad \alpha_n \leq \gamma, \quad \alpha_n \geq 0 \text{ where } \tilde{L}(\boldsymbol{\alpha}) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m \mathbf{x}_n^{\mathrm{T}} \mathbf{x}_m + \sum_{n=1}^{N} \alpha_n$$

and make predictions by $f_{\boldsymbol{\theta}}(\mathbf{x}) = \sum_{n=1}^{N} \alpha_n^* y_n \mathbf{x}^{\mathrm{T}} \mathbf{x}_n + b$

- Observation: instances (both training and new instances after deployment) only enter these equations in the form of dot products
- **Why is that interesting?** Assume we want to have a non-linear classifier, and to this end employ a feature map
  - Assume that the feature map itself is complex and/or high-dimensional
  - But assume that the dot products of the form $\Phi(\mathbf{x})^{\mathrm{T}}\Phi(\overline{\mathbf{x}})$ are easy to compute
  - Using a dual SVM, we never have to explicitly compute $\Phi(\mathbf{x})$!
  - Can use very high-dimensional $\Phi(\mathbf{x})$ as long as we can compute $\Phi(\mathbf{x})^{\mathrm{T}}\Phi(\overline{\mathbf{x}})$

Example: Kernel Trick

- **Example**: we map instances from $\mathbf{x} \in \mathbb{R}^2$ to $\Phi(\mathbf{x}) \in \mathbb{R}^6$ via

$$\Phi(\mathbf{x}) = \begin{pmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix}$$

- Then the inner product $\Phi(\mathbf{x})^T\Phi(\overline{\mathbf{x}})$ can be computed easily as

$$\begin{aligned}\Phi(\mathbf{x})^T\Phi(\overline{\mathbf{x}}) &= 1 + 2x_1\overline{x}_1 + 2x_2\overline{x}_2 + x_1^2\overline{x}_1^2 + x_2^2\overline{x}_2^2 + 2x_1x_2\overline{x}_1\overline{x}_2 \\ &= (1 + x_1\overline{x}_1 + x_2\overline{x}_2)^2\end{aligned}$$

- The function $K(\mathbf{x}, \overline{\mathbf{x}}) = (1 + x_1\overline{x}_1 + x_2\overline{x}_2)^2 = (1 + \mathbf{x}^T\overline{\mathbf{x}})^2$ is called a **kernel function**
- The kernel function operates on $\mathbf{x} \in \mathbb{R}^2$, but computes the dot product $\Phi(\mathbf{x})^T\Phi(\overline{\mathbf{x}})$ after an implicit feature mapping $\Phi(\mathbf{x})$
- The dimensionality of the mapping can be increased by choosing a higher exponent

Kernel Functions

- More formally, a **kernel function** is a function

$$K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

  that is symmetric and positive semidefinite
- Symmetric: $K(\mathbf{x}, \overline{\mathbf{x}}) = K(\overline{\mathbf{x}}, \mathbf{x})$
- Positive semidefinite: for any $N \in \mathbb{N}$, $\mathbf{c}_1, ..., c_N \in \mathbb{R}$, $\mathbf{x}_1, ..., \mathbf{x}_N \in \mathcal{X}$ it has to hold that

$$\sum_{n=1}^{N} \sum_{m=1}^{N} c_n c_m K(\mathbf{x}_n, \mathbf{x}_m) \geq 0$$

- Different kernel functions $K(\mathbf{x}, \overline{\mathbf{x}})$ correspond to different feature maps $\Phi(\mathbf{x})$
- Training a dual SVM and replacing all dot products of the form $\mathbf{x}^{\mathrm{T}} \overline{\mathbf{x}}$ with a kernel function $K(\mathbf{x}, \overline{\mathbf{x}})$ is equivalent to learning a linear SVM function on the transformed instances $\Phi(\mathbf{x})$

Popular Kernel Functions

- Popular examples of kernel functions:
  - Linear kernel: $K(\mathbf{x}, \overline{\mathbf{x}}) = \mathbf{x}^{\mathrm{T}} \overline{\mathbf{x}}$. This is obviously identical to a linear SVM on the original feature representation (as nothing is replaced)
  - Polynomial kernel of degree *d*: $K(\mathbf{x}, \overline{\mathbf{x}}) = (1 + \mathbf{x}^{\mathrm{T}} \overline{\mathbf{x}})^d$. This is equivalent to learning a linear SVM model on data transformed by a polynomial feature map $\Phi(\mathbf{x})$. The dimensionality of the feature map increases with *d*
  - Gaussian kernel (or radial basis function kernel): $K(\mathbf{x}, \overline{\mathbf{x}}) = \exp(-\| \mathbf{x} - \overline{\mathbf{x}} \|^2 / c)$. This is equivalent to a linear model on a feature map $\Phi(\mathbf{x})$ of infinite dimension

Summary: Support Vector Machines

- Support vector machines (or SVMs) are another approach to learn a linear binary classification model
- SVMs attempt to find a linear model by maximizing the margin between the decision boundary and the closest training data points
- If the training data are not linearly separable, slack variables can be introduced to allow some violations of the margin and misclassifications (soft margin SVM)
- In the primal formulation, soft margin SVMs result in a quadratic optimization problem with inequality constraints, or can be expressed as minimizing hinge loss + regularizer
- There is also a dual formulation, where instead of optimizing over a feature vector $\boldsymbol{\theta} \in \mathbb{R}^M$, we optimize over a vector of dual variables $\boldsymbol{\alpha} \in \mathbb{R}^M$ and get sparse solutions
- The dual formulation has the advantage that kernel functions can be employed to obtain a non-linear model by an implicit feature map

Further Reading

- Hastie et al. 2005, Chapter 12.1-3, Murphy 2012 Chapter 14.1+2+5, James et al., Chapter 9

- Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The Elements of Statistical Learning: Data Mining, Inference and Prediction, volume 27. Springer, 2005.
- Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. An Introduction to Statistical Learning. Springer, 2013.
- Kevin P. Murphy. Machine Learning: A Probabilistic Perspective. The MIT Press, 2012.