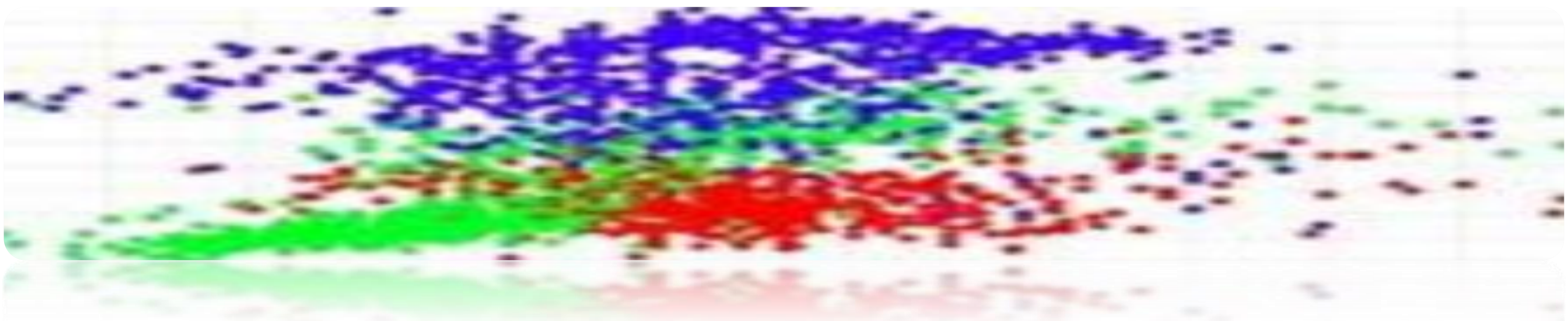
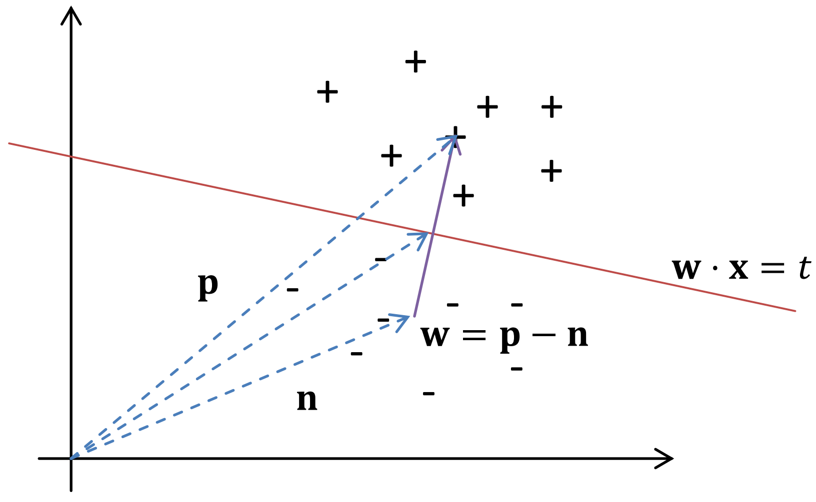


LINEAR CLASSIFICATION MODELS

Outline

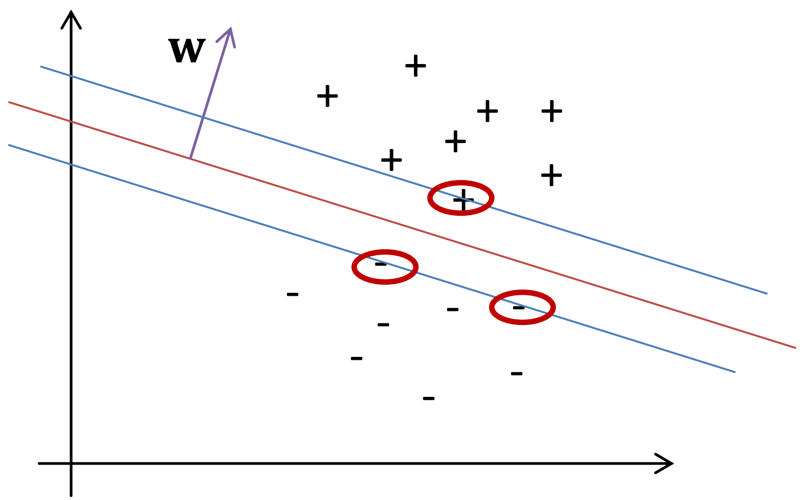
- Geometric classification models
  - Perceptron
  - Winnow
  - Support Vector Machines
- Probabilistic classification models
  - Naïve Bayes
  - Multinomial Naïve Bayes

Geometric classification models



- Basic linear classifier constructs a linear decision boundary $\mathbf{w} \cdot \mathbf{x} = t$
- $\mathbf{w}$ is the vector from negative to positive center
- $\mathbf{x}$ is an instance feature vector to be classified
- In the above model: $t = \frac{(\mathbf{p}-\mathbf{n})(\mathbf{p}+\mathbf{n})}{2} = \frac{\|\mathbf{p}\|^2 - \|\mathbf{n}\|^2}{2}$

Geometric models: Maximum-margin classifiers



- Decision boundary maximizes the margin between negative and positive class
- A geometric model is called translation invariant if it does not depend on the origin of the coordinate system is

Expressiveness of geometric linear models

- General model (i.e., target function)

$$f(\mathbf{x}) = \begin{cases} 1, & w_1 x_1 + \cdots + w_k x_k \geq t \\ -1, & otherwise \end{cases}$$

$$\Leftrightarrow$$

$$f(\mathbf{x}) = sign(\mathbf{w}' \cdot \mathbf{x}'),$$

$$\mathbf{w}' \leftarrow (-t, w_1, \dots, w_k), \qquad \mathbf{x}' = \begin{pmatrix} 1 \\ x_1 \\ \cdot \\ \cdot \\ \cdot \\ x_k \end{pmatrix}$$

- Classes that are expressed as a disjunction of Boolean features i.e., $C = X_1 \vee \cdots \vee X_k$ can separated

- Separation of Exclusive-Or or general DNF representations, e.g., $C = (X_1 \wedge \overline{X_2}) \vee (\overline{X_1} \wedge X_2)$, or $C = (X_1 \wedge X_2) \vee (X_3 \wedge X_4) \vee (X_2 \wedge X_3)$, is not possible

  → Problem if data is not linearly separable

The Perceptron Algorithm

- Invented by F. Rosenblatt in 1957
- For labeled data $(\mathbf{x}_1, l(\mathbf{x}_1)), \dots, (\mathbf{x}_n, l(\mathbf{x}_n))$, $\mathbf{x}_i \in \mathbb{R}^k$, $l(\mathbf{x}_i) \in \{-1,1\}$
- Learn $\mathbf{w}'$ for $f = sign(\mathbf{w}' \cdot \mathbf{x}')$ as follows

1. Initialize $\mathbf{w}'$ to $(0, \dots, 0)$
2. For each $\mathbf{x}_i$

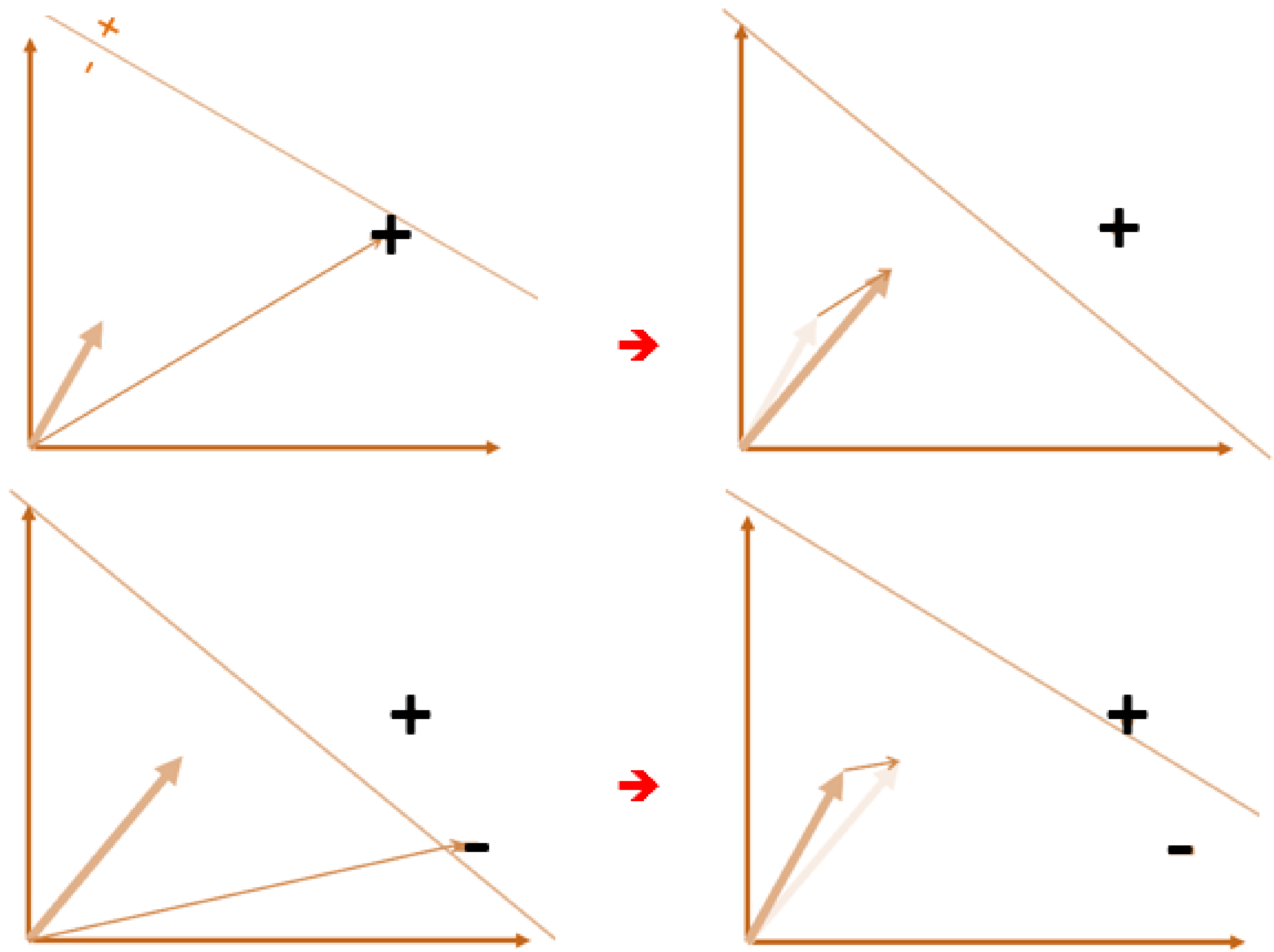
   If $l(\mathbf{x}_i) \neq sign(\mathbf{w}' \cdot \mathbf{x}_i')$ then //e.g., learning rate $r = 0.1$

   $$\mathbf{w}' \leftarrow \mathbf{w}' + r \cdot l(\mathbf{x}_i) \cdot \mathbf{x}_i'$$

3. Repeat 2. until $\frac{1}{n}\sum_{j=1}^{n}\left(l(\mathbf{x}_j) - sign(\mathbf{w}' \cdot \mathbf{x}_j')\right) < \gamma$

   //for user-set threshold γ

Example



Convergence of the Perceptron Algorithm

- Theorem 1
  - If the data is not linearly separable, the algorithm may not finish

- Theorem 2 (Novikoff 1962)
  - Let $\gamma \in \mathbb{R}$ be chosen in such a way that for all labeled instances $(\mathbf{w}' \cdot \mathbf{x}_i{}')l(\mathbf{x}_i) > \gamma$ and let $R = \max(|x_1|, \ldots, |x_{k+1}|; \mathbf{x} \in Training)$.

    The number of all possible updates to $\mathbf{w}'$ by the algorithm is bounded by $\frac{R^2}{\gamma^2}$

Expressiveness of the Perceptron Algorithm

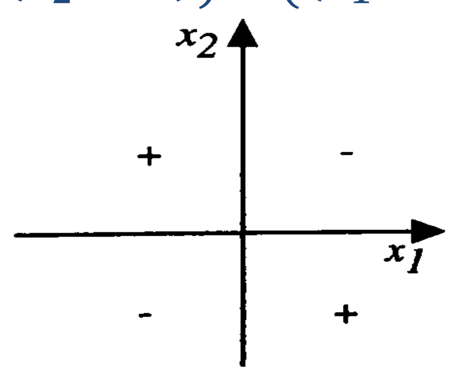
- Theorem

  - Classes that can be represented by primitive Boolean combination of features, (AND, OR, NAND, NOR) can be **represented by the Perceptron Algorithm (i.e., can be separated from their complement)**

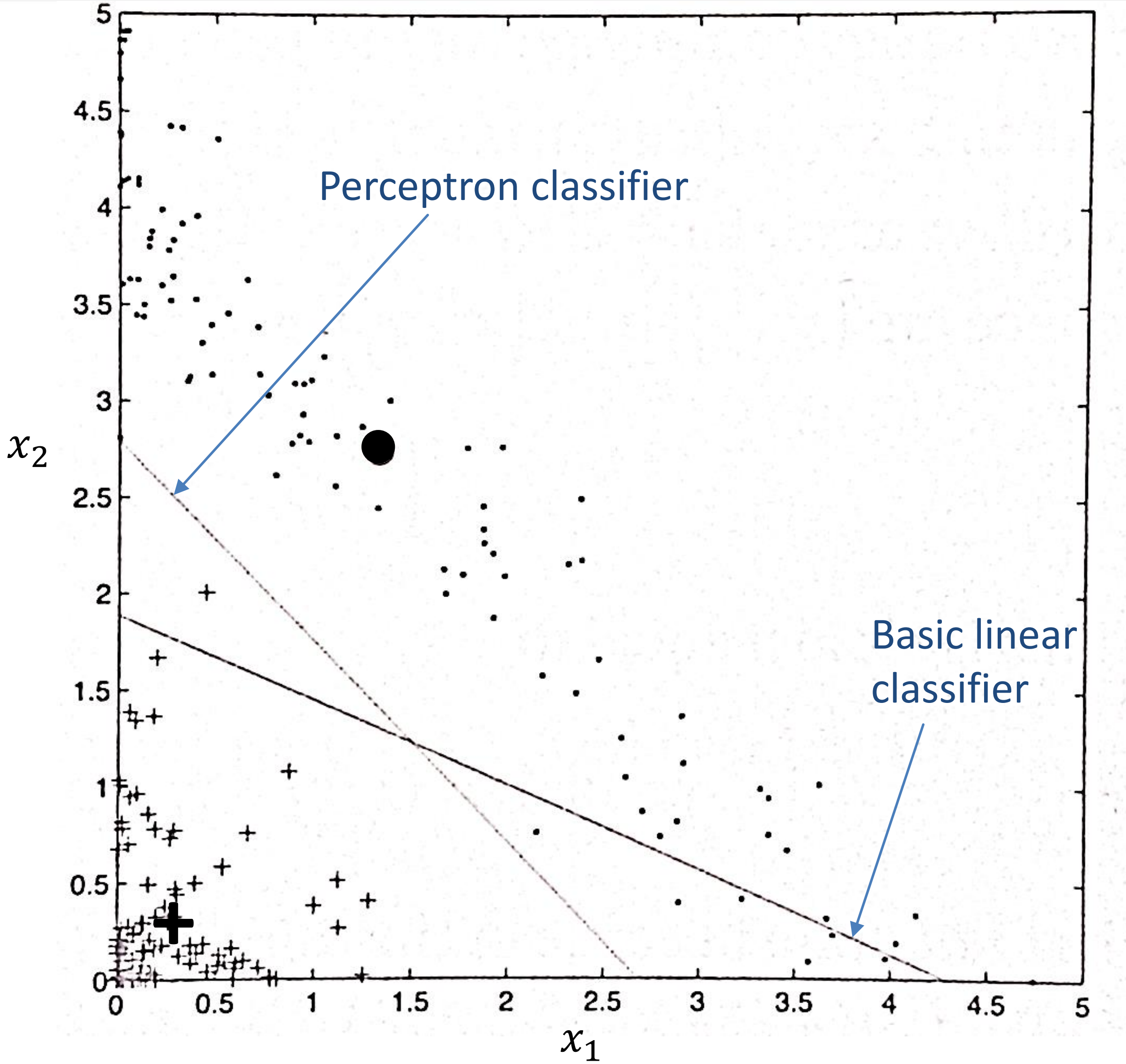
  - Classes represented by XOR feature combinations or more complex formulas (i.e., for which multiple NANDs or NORs are needed) cannot be represented by the Perceptron Algorithm

- Abstract representation of an XOR case (for: $x_1 \in \{a, -a\}, x_2 \in \{b, -b\}$)

$$C^+ = \big((x_1 = -a) \wedge (x_2 = b)\big) \vee \big((x_1 = a) \wedge (x_2 = -b)\big)$$



Basic linear classifier vs. Perceptron algorithm



Source: *Machine Learning* by P. Flach

The Winnow Algorithm

$$f(\mathbf{x}) = \begin{cases} 1, & w_1 x_1 + \cdots + w_k x_k \geq t \\ 0, & otherwise \end{cases}$$

- For labeled data $(\mathbf{x}_1, l(\mathbf{x}_1)), \dots, (\mathbf{x}_n, l(\mathbf{x}_n))$, $\mathbf{x}_i \in \{0,1\}^k$, $l(\mathbf{x}_i) \in \{0,1\}$
- Learn $\mathbf{w}$ for $f(\mathbf{x})$ as follows

1. Set all $w_i \coloneqq 1$, $t \coloneqq \frac{n}{2}$

   //leads to good bounds on the number of possible updates

2. For each example $\mathbf{x}_i$

   If $f(\mathbf{x}_i) = 0 \wedge l(\mathbf{x}_i) = 1$ //promote involved weights

   For each j with $x_j = 1$ set $w_j \leftarrow 2w_j$

   If $f(\mathbf{x}_i) = 1 \wedge l(\mathbf{x}_i) = 0$ //demote involved weights

   For each j with $x_j = 1$ set $w_j \leftarrow \frac{w_j}{2}$

3. Repeat 2. until $\frac{1}{\mathrm{n}} \sum_{j=1}^{n} \left(l(\mathbf{x}_j) - f(\mathbf{x}_j) \right) < \gamma$

Number of updates for the Winnow Algorithm

- Initially $f(\mathbf{x}) = x_{i_1} \vee \cdots \vee x_{i_l}$ for certain $x_{i_j} \neq 0$ (i.e., a monotone Boolean function)

  → We need to update $l \leq k$ weight components

- Also none of these weight components can get greater than t

- The algorithm performs "binary search" in the range of each weight component, which is given by $(0, \frac{n}{2}]$

  → Bound on updates $O(k \log n)$

Advantages of Winnow

- Winnow Algorithm is robust in the presence of label noise or feature noise (important because often training data does not represent well the class distributions)
- Popular in natural language processing where many features are binary
- There exist other robust variations
  - Example: following updates can be used (balanced version)

```
2. For each example x_i
     If (w_j^+ − w_j^−) · x_i < t ∧ l(x_i) = 1  //promote involved weights
           For each j with x_j = 1 set w_j^+ ← 2w_j^+, w_j^− ← w_j^−/2
     If (w_j^+ − w_j^−) · x_i > t ∧ l(x_i) = 0  //demote involved weights
           For each j with x_j = 1 set w_j^+ ← w_j^+/2, w_j^− ← 2w_j^−
```

2. For each example $\mathbf{x}_i$

If $\left(w_j^+ - w_j^-\right) \cdot \mathbf{x}_i < t \wedge l(\mathbf{x}_i) = 1$ //promote involved weights

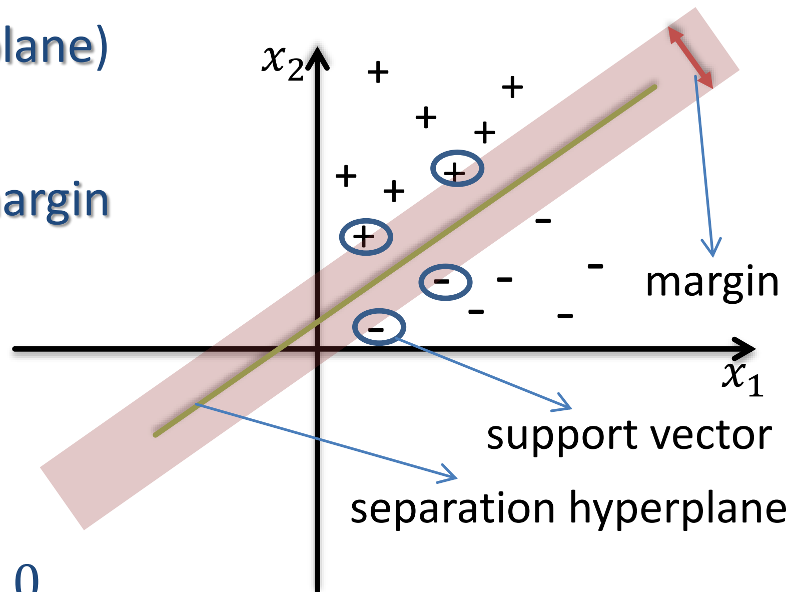
For each j with $x_j = 1$ set $w_j^+ \leftarrow 2w_j^+$, $w_j^- \leftarrow \frac{w_j^-}{2}$

If $\left(w_j^+ - w_j^-\right) \cdot \mathbf{x}_i > t \wedge l(\mathbf{x}_i) = 0$ //demote involved weights

For each j with $x_j = 1$ set $w_j^+ \leftarrow \frac{w_j^+}{2}$, $w_j^- \leftarrow 2w_j^-$

Support Vector Machines

- Goal: Set the separation plane so that margin is maximized (→ **maximum margin hyperplane**)

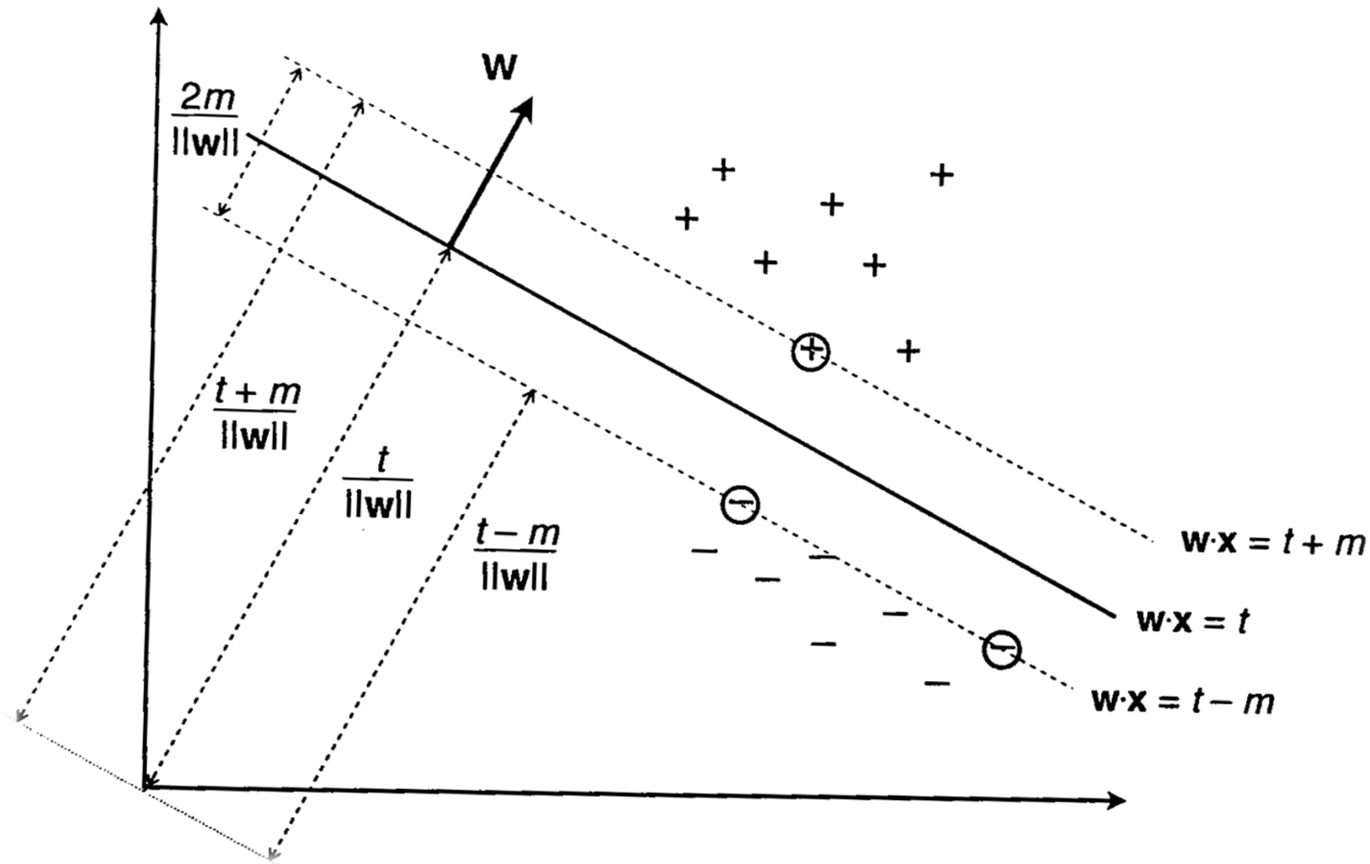


- Linear SVMs are also called **maximum margin classifiers**

- Hyperplane is described by $\mathbf{w} \cdot \mathbf{x}_i - t = 0$, where $\mathbf{w}$ is a normal vector to the hyperplane and $\mathbf{x}$ a point on the hyperplane

- The offset (distance) of the hyperplane from the origin: $\frac{t}{||\mathbf{w}||}$

Geometry of Support Vector Machines



Source: *Machine Learning* by P. Flach

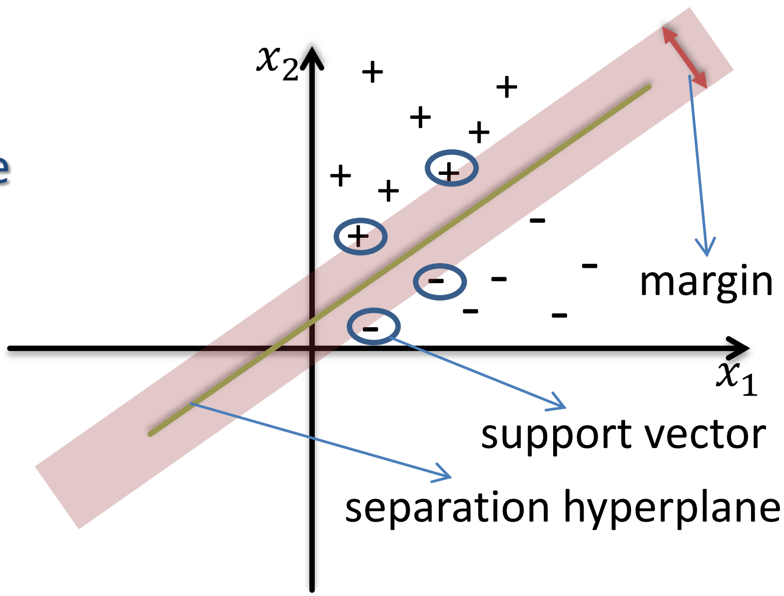
Support Vector Machines: Margin width

- For every data point on the positive side

$$\mathbf{w} \cdot \mathbf{x}_i - t \geq 1$$

- For every data point on the negative side

$$\mathbf{w} \cdot \mathbf{x}_i - t \leq -1$$



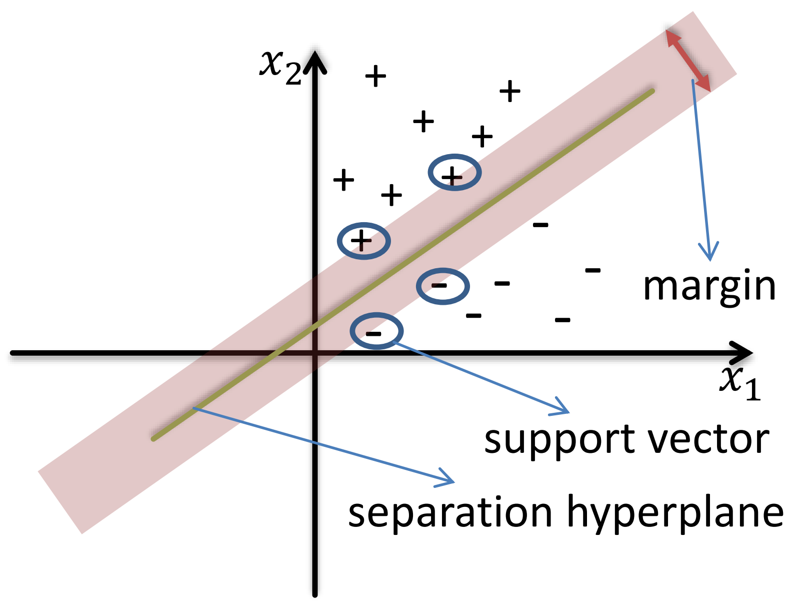
➔ Margin width: $\frac{2}{||\mathbf{w}||}$

➔ Increasing width ⇔ Minimizing $||\mathbf{w}||$

- $||\mathbf{w}||$ is difficult to minimize (it involves square root)

- $\frac{1}{2}||\mathbf{w}||^2$ is easier to minimize.

Optimization problem (1)



- Quadratic programming optimization problem

$$\operatorname{argmin}_{\mathbf{w},t} \ \frac{1}{2}\|\mathbf{w}\|^2$$

subject to $y_i(\mathbf{w} \cdot \mathbf{x}_i - t) \geq 1$

for all trainings instances $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and their true labels $y_1, \ldots, y_n$

Optimization problem (2)

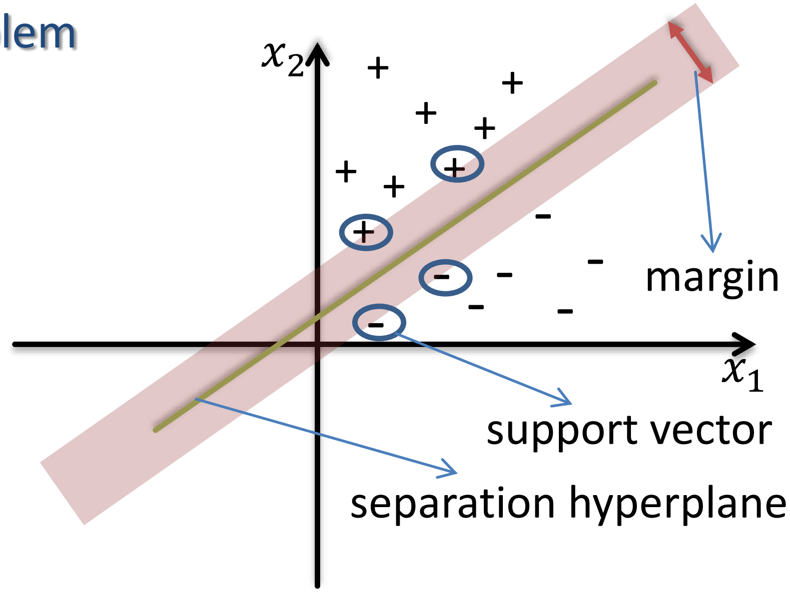
- **Quadratic programming optimization problem**

  $\text{argmin}_{\mathbf{w},t} \ \frac{1}{2}||\mathbf{w}||^2$

  subject to $y_i(\mathbf{w} \cdot \mathbf{x}_i - t) \geq 1$

  for all trainings instances $\mathbf{x}_1, \dots, \mathbf{x}_n$

  and their true labels $y_1, \dots, y_n$



- **Primal form**

$$\boldsymbol{\Lambda}(\mathbf{w}, t, \alpha_1, \dots, \alpha_n) = \left(\frac{1}{2}||\mathbf{w}||^2 - \sum_i \alpha_i (y_i(\mathbf{w} \cdot \mathbf{x}_i - t) - 1\,)\right)$$

$$\underset{\mathbf{w},t}{\text{argmin}} \ \underset{\boldsymbol{\alpha} \geq 0}{\max} \ \boldsymbol{\Lambda}(\mathbf{w}, t, \alpha_1, \dots, \alpha_n)$$

  for Lagrange multipliers $\alpha_i \geq 0, 1 \leq i \leq n$

Note: We are not interested in instances $\mathbf{x}_i$ for which $y_i(\mathbf{w} \cdot \mathbf{x}_i - t) - 1 > 0$, for these points α_i must be set to 0, in order to maximize the expression in $\boldsymbol{\alpha}$

Finding the optimal w

$$\boldsymbol{\Lambda}(\mathbf{w}, t, \alpha_1, \dots, \alpha_n) = \left(\frac{1}{2}\|\mathbf{w}\|^2 - \sum_i \alpha_i (y_i(\mathbf{w}\cdot\mathbf{x}_i - t) - 1\,)\right)$$

$$= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_i \alpha_i y_i (\mathbf{w}\cdot\mathbf{x}_i) + \sum_i \alpha_i y_i t + \sum_i \alpha_i$$

$$= \frac{1}{2}\mathbf{w}\cdot\mathbf{w} - \mathbf{w}\left(\sum_i \alpha_i y_i \mathbf{x}_i\right) + t\left(\sum_i \alpha_i y_i\right) + \sum_i \alpha_i$$

$$\frac{\partial \boldsymbol{\Lambda}(\mathbf{w}, t, \alpha_1, \dots, \alpha_n)}{\partial \mathbf{w}} = 0 \;\Leftrightarrow\; \mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial \boldsymbol{\Lambda}(\mathbf{w}, t, \alpha_1, \dots, \alpha_n)}{\partial t} = 0 \;\Leftrightarrow\; \sum_i \alpha_i y_i = 0$$

By plugging these values back into $\boldsymbol{\Lambda}$ we can get rid of **w** and t

Dual problem

$$\boldsymbol{\Lambda}(\mathbf{w}, t, \alpha_1, \dots, \alpha_n) = \frac{1}{2}\mathbf{w}\cdot\mathbf{w} - \mathbf{w}\left(\sum_i \alpha_i y_i \mathbf{x}_i\right) + t\left(\sum_i \alpha_i y_i\right) + \sum_i \alpha_i$$

$$\frac{\partial \boldsymbol{\Lambda}(\mathbf{w}, t, \alpha_1, \dots, \alpha_n)}{\partial \mathbf{w}} = 0 \Leftrightarrow \mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial \boldsymbol{\Lambda}(\mathbf{w}, t, \alpha_1, \dots, \alpha_n)}{\partial t} = 0 \Leftrightarrow \sum_i \alpha_i y_i = 0$$

By plugging these values back into $\boldsymbol{\Lambda}$ we can get rid of $\mathbf{w}$ and t

- Quadratic optimization problem

$$\boldsymbol{\Lambda}(\alpha_1, \dots, \alpha_n) = -\frac{1}{2}\left(\sum_i \alpha_i y_i \mathbf{x}_i\right)\left(\sum_i \alpha_i y_i \mathbf{x}_i\right) + \sum_i \alpha_i$$

$$\underset{\boldsymbol{\alpha}\geq 0}{\operatorname{argmax}} \sum_i \alpha_i - \frac{1}{2}\left(\sum_i \sum_j \alpha_i \alpha_j y_i y_j \underbrace{\mathbf{x}_i \cdot \mathbf{x}_j}_{\text{kernel}}\right)$$

subject to $\sum_i \alpha_i y_i = 0$ (and $\alpha_i \geq 0, 1 \leq i \leq n$)

Solving the SVM optimization problem

- In general, the dual form of a quadratic optimization problem is only a lower bound of the primal formulation

- Under certain conditions, known as Karush-Kuhn-Tucker conditions, which are satisfied by the dual form of the SVM optimization problem, the two solutions become equal

- In practice the dual form is solved through quadratic programming optimization techniques

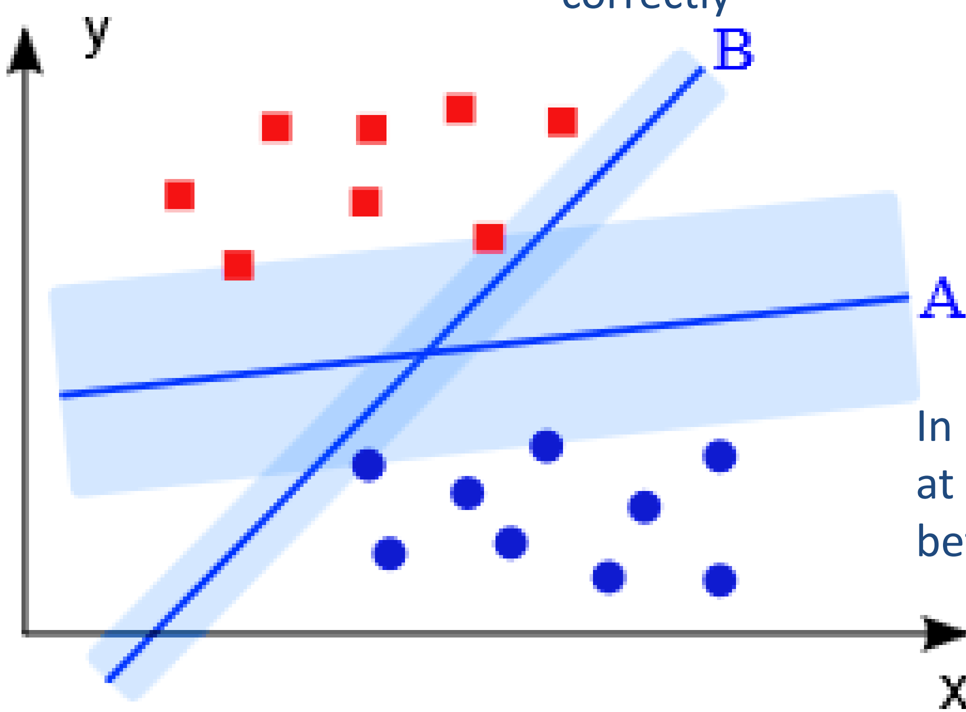
- Once the $\alpha_i, 1 \leq i \leq n$ are known the SVM classifier is:

$$\mathbf{w} \cdot \mathbf{x} + t = \left(\sum_i \alpha_i y_i \mathbf{x}_i\right) \cdot \mathbf{x} + t \begin{cases} \geq 1 & \Rightarrow + \\ \leq -1 & \Rightarrow - \end{cases}$$

- What about t?

SVM vs. Perceptron or Winnow

For the Perceptron or Winnow Algorithm the only goal is to classify every instance correctly

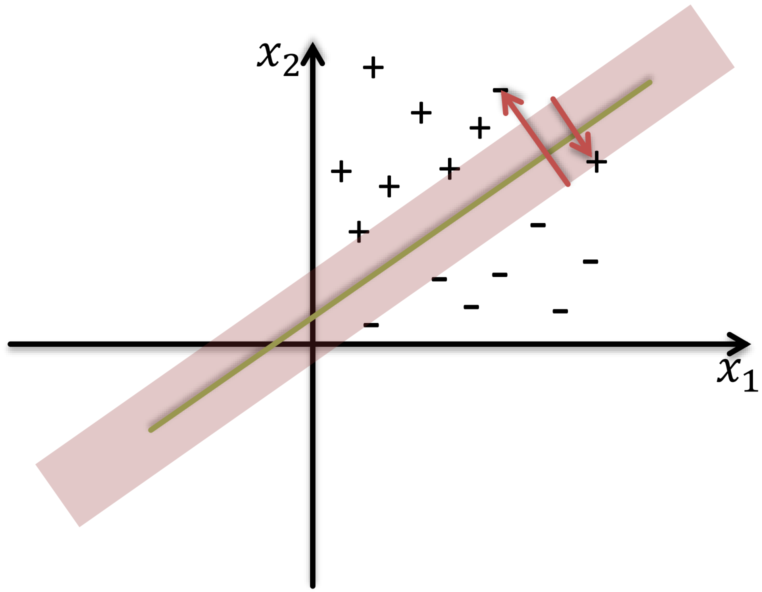


Source: Wikipedia

In addition an SVM aims at maximizing the margin between the two classes

Soft-margin SVMs

- Can the model be extended to handle data that is “almost” linearly separable?



- Primal form

$\text{argmin}_{\mathbf{w},t,\boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i$

subject to $y_i(\mathbf{w}\cdot\mathbf{x}_i - t) \geq 1 - \xi_i, \xi_i \geq 0$

for all trainings instances $\mathbf{x}_1, \dots, \mathbf{x}_n$

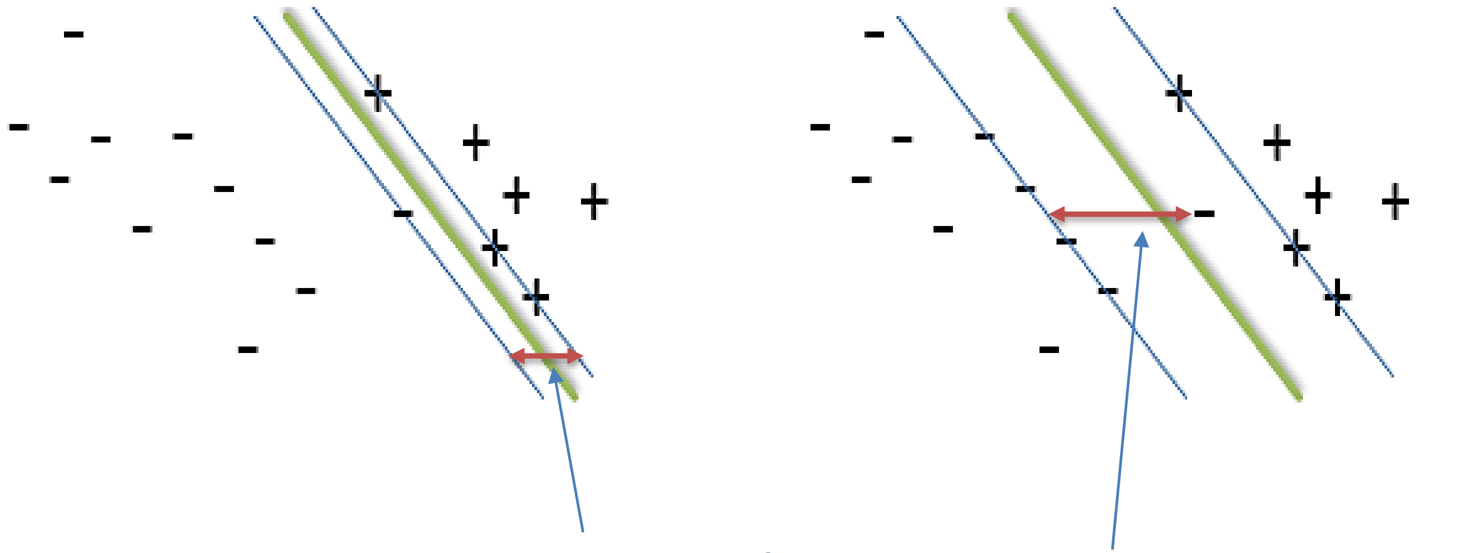
and their true labels $y_1, \dots, y_n$

  - C: is user-specified trade-off between margin width and error
  - ξ_i: slack variables representing real-valued errors

SVM vs. Soft-Margin SVM

SVM

Soft-Margin SVM



Margin is $m = 1 - \xi$

ξ: slack variable

Penalty, also called hinge loss

Dual problem for the Soft-Margin SVM

$$\boldsymbol{\Lambda}(\mathbf{w}, t, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$
$$= \frac{1}{2}\mathbf{w}\cdot\mathbf{w} - \mathbf{w}\left(\sum_i \alpha_i y_i \mathbf{x}_i\right) + t\left(\sum_i \alpha_i y_i\right) + \sum_i \alpha_i + \sum_i (C - \alpha_i - \beta_i)\,\xi_i$$

$$\frac{\partial \boldsymbol{\Lambda}(\mathbf{w}, t, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})}{\partial \mathbf{w}} = 0 \iff \mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial \boldsymbol{\Lambda}(\mathbf{w}, t, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})}{\partial t} = 0 \iff \sum_i \alpha_i y_i = 0$$

$$\frac{\partial \boldsymbol{\Lambda}(\mathbf{w}, t, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})}{\partial \xi_i} = 0 \iff \forall i\colon\ C - \alpha_i - \beta_i = 0 \iff_{\boldsymbol{\beta}\geq\mathbf{0}} C \geq \alpha_i \geq 0$$

- Dual problem

$$\underset{\boldsymbol{\alpha}\geq 0}{\operatorname{argmax}} \sum_i \alpha_i - \frac{1}{2}\left(\sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j\right)$$

subject to $\sum_i \alpha_i y_i = 0$ (and $C \geq \alpha_i \geq 0, 1 \leq i \leq n$)

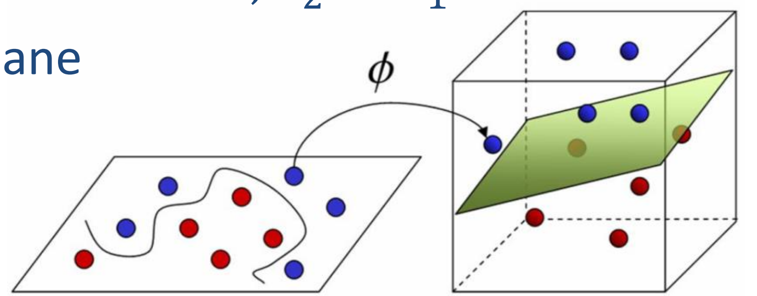
Summary of SVMs

- Come in different variations (principle is the same)
- Simultaneously minimize the empirical error and maximize the margin width
- Same expressiveness as the perceptron algorithm

- In general, quadratic training time (quadratic optimization)
  - T. Joachims showed linear training time for linear SVMs (*www.joachims.org/publications/joachims_06a.pdf*)
- Directly applicable only to two-class problems
- Parameter values in a solution are difficult to interpret

Kernel Trick

- Apply non-linear transformation function $\phi: \mathbb{R}^{d_1} \mapsto \mathbb{R}^{d_2}, d_2 > d_1$ to input instances $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and find hyperplane $\mathbf{w} \cdot \phi(\mathbf{x}) = t$ that separates the classes



- Same optimization problem as before:

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \left(\sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \boldsymbol{\kappa}(\mathbf{x}_i, \mathbf{x}_j) \right)$$

subject to $\alpha_i > 0, \forall \alpha_i$ and $\sum_i \alpha_i y_i = 0$

- Classification through

$$\mathbf{w} \cdot \mathbf{x} - t = \sum_i \alpha_i y_i \boldsymbol{\kappa}(\mathbf{x}_i \cdot \mathbf{x}) - t \begin{cases} \geq 1 & \Rightarrow + \\ \leq -1 & \Rightarrow - \end{cases}$$

- Typical kernel functions:
  - $\boldsymbol{\kappa}(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2 + c)^d$ (polynomial kernel)
  - $\boldsymbol{\kappa}(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}{2\sigma^2}\right)$ (Gaussian kernel)

Kernel trick example

- Suppose that $\mathbf{n} = (0,0)$ and $\mathbf{p} = (0,1)$
- Let us further suppose that $\mathbf{p}$ has been derived from two positive examples, i.e., $\mathbf{p} = \frac{1}{2}(\mathbf{p}_1 + \mathbf{p}_2)$ with $\mathbf{p}_1 = (-1,1)$ and $\mathbf{p}_2 = (1,1)$
- For the basic linear classifier, the separation hyperplane was defined as

$$(\mathbf{p} - \mathbf{n}) \cdot \mathbf{x} = t \Leftrightarrow \frac{1}{2}(\mathbf{p}_1 \cdot \mathbf{x} + \mathbf{p}_2 \cdot \mathbf{x}) - \mathbf{n} \cdot \mathbf{x} = t$$

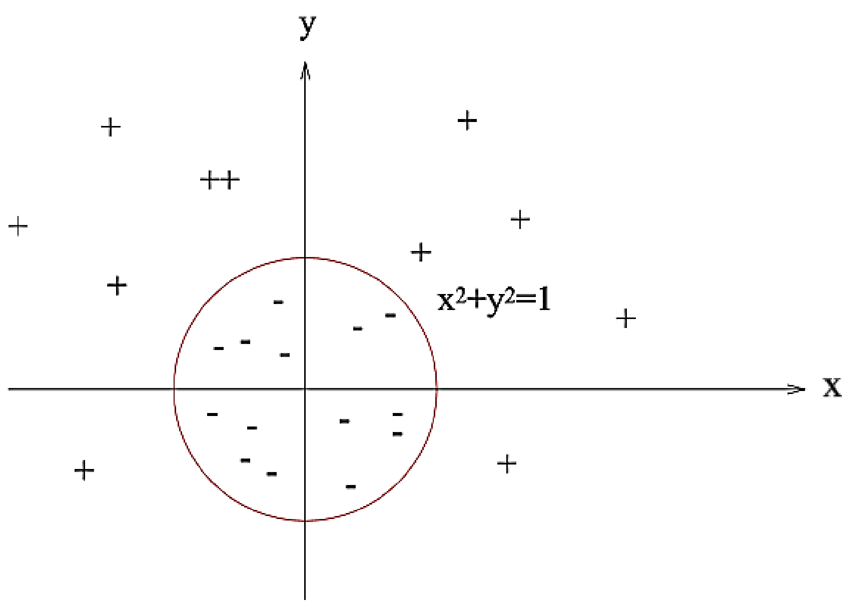
- Applying the kernel trick

$$\mathbf{x} = (x, y) \mapsto \mathbf{x}' = \left(x^2, y^2, \sqrt{2}xy\right)$$

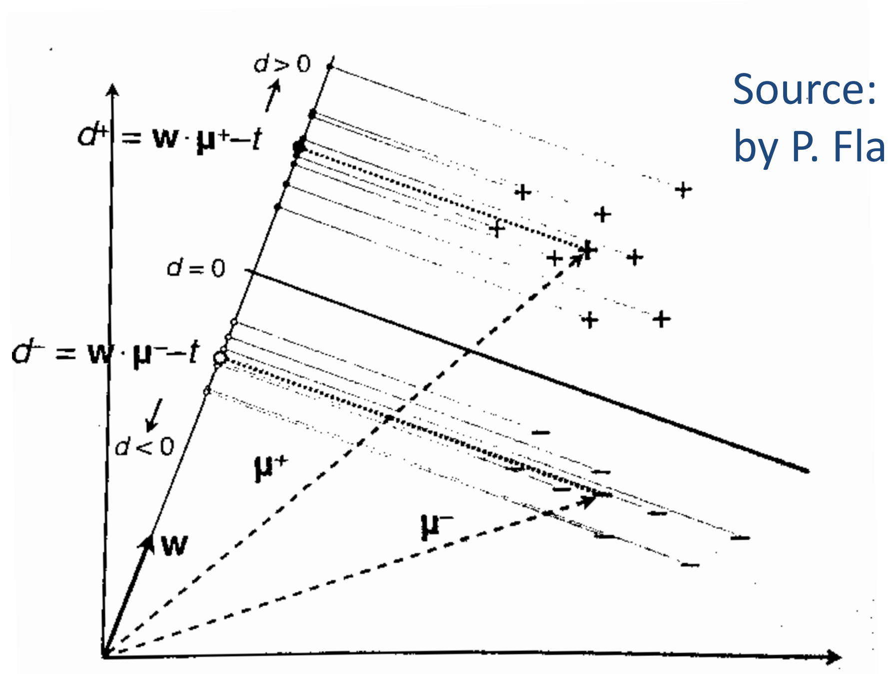
$$\begin{aligned} \boldsymbol{\kappa}(\mathbf{x}_1 \cdot \mathbf{x}_2) &= \mathbf{x}_1' \cdot \mathbf{x'}_2 \\ &= x_1^2 x_2^2 + y_1^2 y_2^2 + 2x_1 y_1 x_2 y_2 \\ &= (\mathbf{x}_1 \cdot \mathbf{x}_2)^2 \end{aligned}$$

$$\frac{1}{2}\boldsymbol{\kappa}(\mathbf{p}_1 \cdot \mathbf{x}) + \frac{1}{2}\boldsymbol{\kappa}(\mathbf{p}_2 \cdot \mathbf{x}) - \boldsymbol{\kappa}(\mathbf{n} \cdot \mathbf{x}) = t$$

$\Leftrightarrow x^2 + y^2 = t$ (separation function)



Probabilistic classifiers from linear classifiers



Source: *Machine Learning* by P. Flach

- We can define $\|\mathbf{w}\|$ as the unit length, then

$$d(\mathbf{x}_i) = \frac{\mathbf{w} \cdot \mathbf{x}_i + t}{\|\mathbf{w}\|}$$

- We can learn a (Gaussian) mixture model based on the distances and use Bayes' theorem to derive

$$P(+|d(\mathbf{x}_i)) = \frac{P(d(\mathbf{x}_i)|+)P(+)}{P(d(\mathbf{x}_i)|+)P(+) + P(d(\mathbf{x}_i)|-)P(-)}$$

Naïve Bayes (1)

- $\mathbf{D} = \{\mathbf{d}_1, \dots, \mathbf{d}_n\}$ a corpus of documents, where each document is seen as a set of words, and can be represented as a binary vector

$$\mathbf{d}_i = (w_{i1}, \dots, w_{im}), w_{ij} = \begin{cases} 1, w_j \in \mathbf{d}_i \\ 0, w_j \notin \mathbf{d}_i \end{cases}$$

- $c_1, \dots, c_k$ topics/classes to which a document can belong
- $\mathbf{V} = \{w \in \mathbf{d} | \mathbf{d} \in \mathbf{D}\} = \{w_1, \dots, w_m\}$ vocabulary of all terms in the corpus
- What is the most probable topic for a document?

- We can compute the joint probability of a document $\mathbf{d}_i$ and a topic c_j as

$$P(\mathbf{d}_i, c_j) = P(\mathbf{d}_i | c_j) P(c_j) = P(w_{i1}, \dots, w_{im} | c_j) P(c_j)$$

- Most probable topic for $\mathbf{d}_i$ can be found by computing the maximum a posteriori (MAP)

$$\underset{c_j}{\operatorname{argmax}} \underbrace{P(w_{i1}, \dots, w_{im} | c_j)} P(c_j)$$

Impossible to estimate!!!
Curse of dimensionality

Naïve Bayes (2)

- If we assume independence between the terms given the topic

$$\underset{c_j}{\operatorname{argmax}} P(w_{i1}, \dots, w_{im} | c_j) P(c_j) = \underset{c_j}{\operatorname{argmax}} P(w_{i1} | c_j) \cdot \dots \cdot P(w_{im} | c_j) P(c_j)$$

- Assuming that words follow a multivariate Bernoulli distribution (i.e., categorical distribution) for a given topic

$$P(w | \boldsymbol{\theta}_j) = \prod_{w' \in c_j} P(w' | c_j)^{[\![w = w']\!]}$$

the maximum likelihood estimation of $P(w_{il} | c_j)$ is given by

$$P(w_{il} | c_j) = \frac{\#(w_{il} \wedge c_j)}{\sum_{w \in \mathbf{V}} \#(w \wedge c_j)}, \text{ i.e., fraction of occurrences of } w_{il} \text{ in } c_j$$

- Similar reasoning for topic distribution yields

$$P(c_j) = \frac{\sum_{\mathbf{d} \in \mathbf{D}} [\![c_j \wedge \mathbf{d}]\!]}{\sum_c \sum_{\mathbf{d} \in \mathbf{D}} [\![c \wedge \mathbf{d}]\!]}, \text{ i.e., fraction of occurrences of } c_j \text{ in } \mathbf{D}$$

Naïve Bayes: Smoothing

$$\underset{c_j}{\operatorname{argmax}} P(w_{i1}, \dots, w_{im}|c_j)P(c_j) = \underset{c_j}{\operatorname{argmax}} P(w_{i1}|c_j) \cdot \dots \cdot P(w_{im}|c_j)P(c_j)$$

- What if w_{il} does not occur in topic c_j?
- Use smoothing
  - Laplace smoothing

$$P(w_{il}|c_j) = \frac{\#(w_{il}, c_j) + \alpha}{\sum_{w \in \mathbf{V}}(\#(w, c_j) + \alpha)}$$

  - Jelinek-Mercer smoothing

$$P(w_{il}|c_j) = \lambda P(w_{il}|c_j) + (1-\lambda)P(w_{il}|\mathbf{D})$$

  - Dirichlet smoothing (the larger the topic the lower the smoothing)

$$P(w_{il}|c_j) = \frac{\#(w_{il}, c_j) + \mu P(w_{il}|\mathbf{D})}{\mu + \sum_{w \in \mathbf{V}} \#(w, c_j)}$$

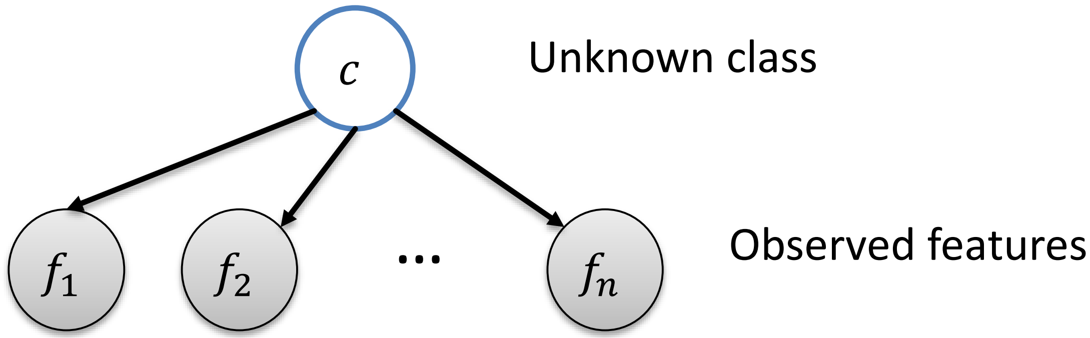
Example: Parameter estimation for Naïve Bayes

Source: *Machine Learning* by P. Flach

	Terms			
E-mail	$a?$	$b?$	$c?$	Class
e_1	0	1	0	+
e_2	0	1	1	+
e_3	1	0	0	+
e_4	1	1	0	+
e_5	1	1	0	–
e_6	1	0	1	–
e_7	1	0	0	–
e_8	0	0	0	–

- Smoothed parameter estimation for positive class with Laplace smoothing with $\alpha = 1 : \boldsymbol{\theta}^{+} = \left(\frac{3}{9}, \frac{4}{9}, \frac{2}{9}\right)$
- Smoothed parameter estimation for the negative class: $\boldsymbol{\theta}^{-} = \left(\frac{4}{8}, \frac{2}{8}, \frac{2}{8}\right)$

Naïve Bayes as a generative model



- Features are generates by some class
- Every feature is independent of the other features given the class
- In such cases, the joint probability $P(f_1, \dots, f_n, c)$ is of interest, because

$$P(c|f_1, \dots, f_n) = \frac{P(f_1, \dots, f_n|c)P(c)}{P(f_1, \dots, f_n)}$$

This term is not defined by the above model, we only know $P(f_1, \dots, f_n|c)P(c)$

Multinomial Naïve Bayes

- In the previous model, we dismissed multiple occurrences of words
- Assuming that documents follow a multinomial distribution

$$P(\mathbf{d}_i|\boldsymbol{\theta}_j) = |\mathbf{d}_i|! \cdot \prod_{w_{il} \in \mathbf{d}_i} \frac{P(w_{il}|c_j)^{freq(w_{il};\mathbf{d}_i)}}{freq(w_{il};\mathbf{d}_i)!}$$

we can estimate the maximum likelihood of $P(w_{il}|c_j)$ as

$$P(w_{il}|c_j) = \frac{\sum_{\mathbf{d} \in c_j} freq(w_{il};\mathbf{d})}{\sum_{w \in \mathbf{V}} freq(w;c_j)}$$

(all previous smoothing strategies can be used)

- Similarly to before we estimate $P(c_j)$ as the fraction of occurrences of c_j in $\mathbf{D}$

Example: Prameters for the Multinomial Naïve Bayes

	Terms			
E-mail	$a?$	$b?$	$c?$	Class
e_1	0	1	0	+
e_2	0	1	1	+
e_3	1	0	0	+
e_4	1	1	0	+
e_5	1	1	0	–
e_6	1	0	1	–
e_7	1	0	0	–
e_8	0	0	0	–

	Terms			
E-mail	$\#a$	$\#b$	$\#c$	Class
e_1	0	3	0	+
e_2	0	3	3	+
e_3	3	0	0	+
e_4	2	3	0	+
e_5	4	3	0	–
e_6	4	0	3	–
e_7	3	0	0	–
e_8	0	0	0	–

Source: *Machine Learning* by P. Flach

- Smoothed parameter estimation for positive class with Laplace smoothing with $\alpha = 1$: $\boldsymbol{\theta}^{+} = \left(\frac{6}{20}, \frac{10}{20}, \frac{4}{20}\right)$
- Smoothed parameter estimation for the negative class: $\boldsymbol{\theta}^{-} = \left(\frac{12}{20}, \frac{4}{20}, \frac{4}{20}\right)$

Naïve Bayes vs. Multinomial Naïve Bayes

	Terms			
E-mail	$a?$	$b?$	$c?$	Class
e_1	0	1	0	+
e_2	0	1	1	+
e_3	1	0	0	+
e_4	1	1	0	+
e_5	1	1	0	−
e_6	1	0	1	−
e_7	1	0	0	−
e_8	0	0	0	−

$\boldsymbol{\theta}^+ = \left(\frac{3}{9}, \frac{4}{9}, \frac{2}{9}\right)$ $\quad \boldsymbol{\theta}^- = \left(\frac{4}{8}, \frac{2}{8}, \frac{2}{8}\right)$

	Terms			
E-mail	$\#a$	$\#b$	$\#c$	Class
e_1	0	3	0	+
e_2	0	3	3	+
e_3	3	0	0	+
e_4	2	3	0	+
e_5	4	3	0	−
e_6	4	0	3	−
e_7	3	0	0	−
e_8	0	0	0	−

$\boldsymbol{\theta}^+ = \left(\frac{6}{20}, \frac{10}{20}, \frac{4}{20}\right)$ $\quad \boldsymbol{\theta}^- = \left(\frac{12}{20}, \frac{4}{20}, \frac{4}{20}\right)$

- Assume we see new document $\mathbf{d} = (\#a = 3, \#b = 1, \#c = 0)$
  - Naïve Bayes: $P(\mathbf{d}|\boldsymbol{\theta}^+)P(\boldsymbol{\theta}^+) = \frac{3}{9} \cdot \frac{4}{9} \cdot \left(1 - \frac{2}{9}\right) \cdot 0.5 = 0.057$ and

    $P(\mathbf{d}|\boldsymbol{\theta}^-)P(\boldsymbol{\theta}^-) = \frac{4}{8} \cdot \frac{2}{8} \cdot \left(1 - \frac{2}{8}\right) \cdot 0.5 = 0.047$
  - Multinomial Naïve Bayes: $P(\mathbf{d}|\boldsymbol{\theta}^+)P(\boldsymbol{\theta}^+) = 4! \cdot \frac{\left(\frac{6}{20}\right)^3}{3!} \cdot \frac{\left(\frac{10}{20}\right)^1}{1!} \cdot \frac{\left(\frac{4}{20}\right)^0}{0!} \cdot 0.5 = 0.027$

    and $P(\mathbf{d}|\boldsymbol{\theta}^-)P(\boldsymbol{\theta}^-) = 4! \cdot \frac{\left(\frac{12}{20}\right)^3}{3!} \cdot \frac{\left(\frac{4}{20}\right)^1}{1!} \cdot \frac{\left(\frac{4}{20}\right)^0}{0!} \cdot 0.5 = 0.0864$

HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Summary of Naïve Bayes models

- Fairly good performance even when independence assumption does not hold
- Can easily handle the dimensionality problem
- Needs relatively few training samples to estimate probabilities (misestimations do not hurt the final calculation of odds $\frac{P(f_1,\ldots,f_n|c)P(c)}{P(f_1,\ldots,f_n|\bar{c})P(\bar{c})}$)
- Performance (e.g. for text classification) is slightly worse than the performance of SVMs or neural networks, but NB methods are much simpler and more efficient
- When features are highly dependent on each other, performance can degrade notably (due to independence assumption)
- Multinomial Naïve Bayes shows empirically better performance than Naïve Bayes on large corpora with high variability in document lengths