Klassifikation

Überblick

- Grundkonzepte
- Entscheidungsbäume
- Evaluierung von Klassifikatoren
- Lernen von Regeln
- Klassifikation mittels Assoziationsregeln
- Naïver Bayescher Klassifikator
- Naïve Bayes für Text Klassifikation
- **Support Vektor Maschinen**
- Ensemble-Methoden: Bagging und Boosting
- Zusammenfassung

Einführung

- Support-Vektor-Maschinen wurden von V. Vapnik und Mitarbeitern in den 1970ern Russland entwickelt und wurden nach 1992 im Westen bekannt.
- SVMs sind lineare Klassifikatoren, die eine Hyperebende bestimmen, um zwei Klassen zu trennen.
- Kernel-Funktionen werden für nichtlineare Separation genutzt.
- SVM haben eine gut begründete theoretische Basis und sind oft genauer als andere Klassifikatoren, besonders bei hoch-dimensionalen Daten.
- Sie sind sehr für Text-Klassifikation geeignet.

Grundkonzepte

- Die Menge der Trainingsbeispiele *D*

  $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_r, y_r)\}$,

  mit $\mathbf{x}_i = (x_1, x_2, \ldots, x_n)$ ein **Vektor mit beschreibenden numerischen Attributen** aus $X \subseteq R^n$ und y_i das **Klassenattribut** (Ausgabenwert), $y_i \in \{1, -1\}$.

  1: positive Klasse und -1: negative Klasse.
- SVM bestimmt lineare Funktion der Form (**w**: Gewichtsvektor)

$$f(\mathbf{x}) = \langle \mathbf{w} \cdot \mathbf{x} \rangle + b$$

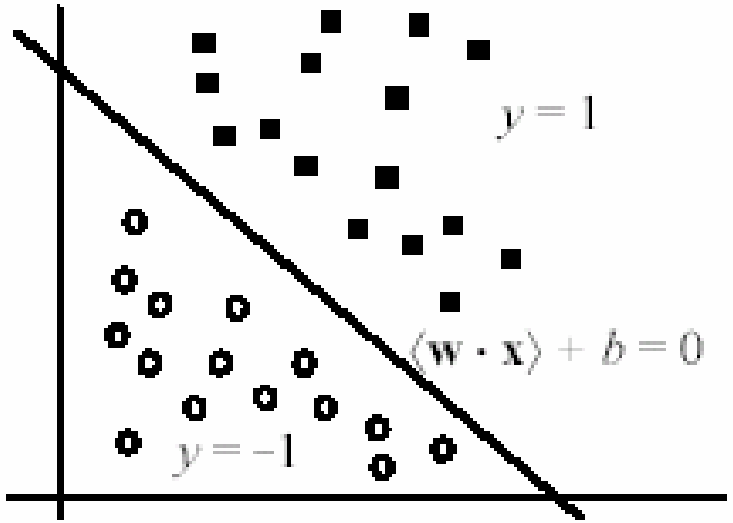
$$y_i = \begin{cases} 1 & if \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 0 \\ -1 & if \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b < 0 \end{cases}$$

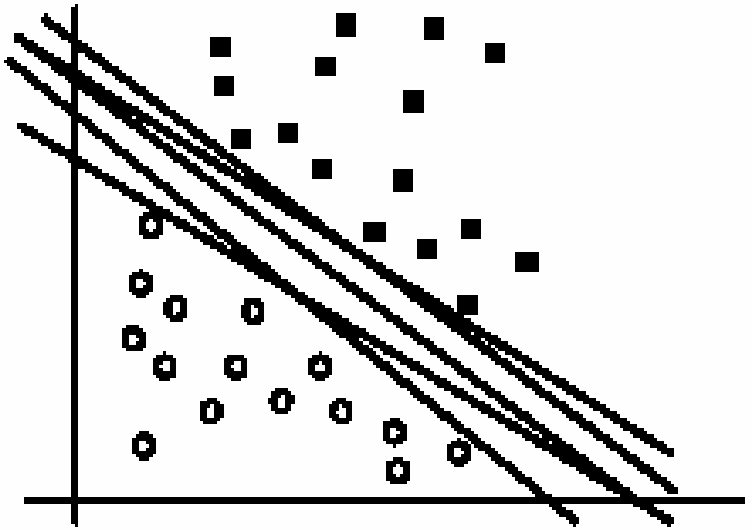
Hyperebene

- Hyperebene separiert positive und negative Trainingsdaten, definiert durch

$$\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$$

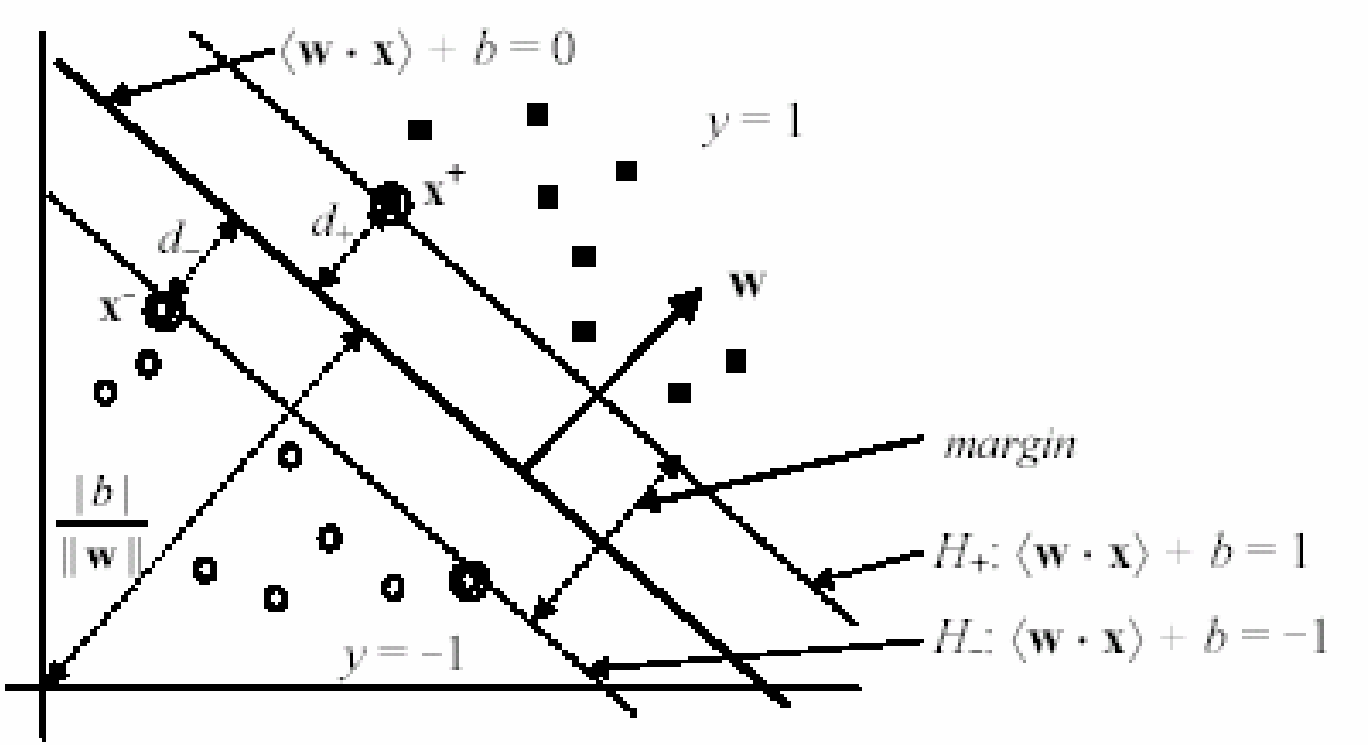
- Auch Entscheidungsgrenze genannt.
- Es gibt viele mögliche Hyperebenen, welche soll gewählt werden?





Hyperebene mit maximalem Rand

- SVM sucht die separierende Hyperebene mit dem größten Rand.
- Theorie zeigt, das diese Hyperebene den Fehler minimiert



Lineare SVM: Separierbarer Fall

- Angenommen die Daten sind linear separierbar.
- Betrachte einen positiven Datenpunkt ($\mathbf{x}^+$, 1) und einen negativen ($\mathbf{x}^-$, -1), die am nächsten zu der Hyperebene liegen $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$.
- Definiere zwei parallele Hyperebenen H_+ und H_-, die jeweils durch $\mathbf{x}^+$ und $\mathbf{x}^-$ laufen. H_+ und H_- sind auch parallel zu $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$.

$$H_+: \quad \langle \mathbf{w} \cdot \mathbf{x}^+ \rangle + b = 1$$

$$H_-: \quad \langle \mathbf{w} \cdot \mathbf{x}^- \rangle + b = -1$$

$$\text{such that} \quad \begin{cases} \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 1 & \text{if } y_i = 1 \\ \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \leq -1 & \text{if } y_i = -1, \end{cases}$$

Berechnung des Randes

- Der Abstand zwischen den zwei Rand-Hyperebenen H_+ und H_- wird **Rand** genannt ($d_+ + d_-$ in der Abbildung, Folie 5).
- Die **Distanz** von $\mathbf{x}_i$ zu der Hyperebene $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$ beträgt senkrecht gemessen:

$$\frac{|\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b|}{\|\mathbf{w}\|} \tag{36}$$

wobei $\|\mathbf{w}\|$ die Norm von $\mathbf{w}$ ist,

$$\|\mathbf{w}\| = \sqrt{<\mathbf{w} \cdot \mathbf{w}>} = \sqrt{w_1^2 + w_2^2 + \ldots + w_n^2} \tag{37}$$

Berechnung des Randes (Forts. …)

- Berechnung von d_+.
  - Statt die Distanz von $\mathbf{x}^+$ zur separierenden Hyperebene $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$ zu berechnen, wird ein beliebiger Punkt $\mathbf{x}_s$ auf $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$ genommen und die Distanz von $\mathbf{x}_s$ zu $\langle \mathbf{w} \cdot \mathbf{x}^+ \rangle + b = 1$ berechnet, durch Anwendung von Gl. (36) und $\langle \mathbf{w} \cdot \mathbf{x}_s \rangle + b = 0$
  - Ebenso die Distanz zu $\langle \mathbf{w} \cdot \mathbf{x}^- \rangle + b = -1$

$$d_+ = \frac{|\langle \mathbf{w} \cdot \mathbf{x_s} \rangle + b - 1|}{\| \mathbf{w} \|} = \frac{1}{\| \mathbf{w} \|} \tag{38}$$

$$margin = d_+ + d_- = \frac{2}{\| \mathbf{w} \|} \tag{39}$$

Optimierungsproblem

Definition (Lineare SVM: separierbarer Fall):

Gegeben eine Menge von linear separiebarer Trainingsbeispiele,

$$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_r, y_r)\}$$

Löse das Minimierungsproblem mit Nebenbeding.,

$$\text{Minimiere}: \frac{\langle \mathbf{w} \cdot \mathbf{w} \rangle}{2} \tag{40}$$

$$\text{Nebenbedingungen}: y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1, \quad i = 1, 2, ..., r$$

$y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 1, \quad i = 1, 2, ..., r$ beschreiben zusammengefaßt

$\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 1$ for $y_i = 1$

$\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \leq -1$ for $y_i = -1$.

Lösung der Minimierung mit Nebenbedingungen

- Standard Lagrange-Methode

$$L_P = \frac{1}{2}\langle \mathbf{w} \cdot \mathbf{w} \rangle - \sum_{i=1}^{r} \alpha_i [y_i (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1] \qquad (41)$$

  wobei $\alpha_i \geq 0$ die **Lagrange-Multiplikatoren** sind.
- Optimierungstheorie sagt, dass eine optimale Lösung für (41) die **Kuhn-Tucker Bedingungen** erfüllen muß, die notwendig aber nicht hinreichend sind

Kuhn-Tucker Bedingungen

$$\frac{\partial L_P}{\partial w_j} = w_j - \sum_{i=1}^{r} y_i \alpha_i \mathbf{x}_i = 0, \quad j = 1, 2, \ldots, m \tag{48}$$

$$\frac{\partial L_P}{\partial b} = -\sum_{i=1}^{r} y_i \alpha_i = 0 \tag{49}$$

$$y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) - 1 \geq 0, \quad i = 1, 2, \ldots, r \tag{50}$$

$$\alpha_i \geq 0, \quad i = 1, 2, \ldots, r \tag{51}$$

$$\alpha_i (y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) - 1) = 0, \quad i = 1, 2, \ldots, r \tag{52}$$

- Gl. (50) ist die Originalmenge der Bedingungen.
- Die komplementären Bedingungen (52) zeigen, dass nur die Punkte auf den Randhyperebenen (H_+ und H_-) können $\alpha_i > 0$ haben, weil nur für sie gilt $y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 = 0$.
- Die Datenpunkte heißen **Support-Vektoren**, alle anderen Parameter $\alpha_i = 0$.

Lösung des Problems

- Im Allgemeinen sind Kuhn-Tucker Bedingungen notwendig für eine optimale Lösung, aber nicht hinreichend.
- Für das vorliegende Minimierungsproblem mit konvexer Zielfunktion und lineare Bedingungen, sind Kuhn-Tucker Bedingungen **notwendig** und **hinreichend** für eine optimale Lösung.
- Die Lösung des Optimierungsproblems ist immer nicht kompliziert wegen der Ungleichungen in den Nebenbedingungen.
- Jedoch, die Lagrange-Methodde für das konvexe Optimierungproblem führt zu einer alternativen **dualen** Formulierung des Problems, welche sich leichter lösen läßt als das Originalproblem (genannt **Primäres Probl.**).

Duale Formulierung

- Vom Primären zum Dualen: Setze alle partiellen Ableitungen der Lagrange-Funktion (41) Null, bezüglich der **primären Variables** (**w** und *b*), und substituiere die Ergebnisse wieder in die Lagrange-Funktion.
  - Substituiere (48) und (49) in die originale Lagrange-Funktion (41) um die primären Variablen zu eliminieren

$$L_D = \sum_{i=1}^{r} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{r} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle, \qquad (55)$$

Duales Optimierungsprolem

$$\text{Maximize: } L_D = \sum_{i=1}^{r} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{r} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle. \tag{56}$$

$$\text{Subject to: } \sum_{i=1}^{r} y_i \alpha_i = 0$$

$$\alpha_i \geq 0, \quad i = 1, 2, \ldots, r.$$

- Für konvexe Zielfunktion und lineare Bedingungen im primären Problem hat das Maximum von L_D die gleichen Werten **w**, b und α_i wie das Minimum von L_P (Primär).
- Lösung von (56) erfordert numerische Techniken, die hier nicht diskutiert werden

Entscheidungsgrenze

- Nach der Lösung von (56), kennt man die Werte für α_i, die genutzt werden um **w** und *b* mittels (48) und (52) zu bestimmen.
- **Entscheidungsgrenze**

$$\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = \sum_{i \in sv} y_i \alpha_i \langle \mathbf{x}_i \cdot \mathbf{x} \rangle + b = 0 \qquad (57)$$

- **Testen**: Nutze (57) geben eine Test-Instanz **z**,

$$sign(\langle \mathbf{w} \cdot \mathbf{z} \rangle + b) = sign\left(\sum_{i \in sv} \alpha_i y_i \langle \mathbf{x}_i \cdot \mathbf{z} \rangle + b \right) \qquad (58)$$

- Fall (58) eine 1 liefert, ist die Test-Instanz **z** positiv; sonst negativ.

Lineare SVM: Nicht-separierbarer Fall

- Lineare separierbare Fall ist die ideale Situation.
- Reale Daten können Rauschen und Ausreißer enthalten.
  - Klassen-label ist falsch oder zufällig in der Anwendung.
- Im separierbarem Fall war das Problem

$$\text{Minimiere}: \quad \frac{\langle \mathbf{w} \cdot \mathbf{w} \rangle}{2}$$

$$\text{Nebenbedingungen}: \quad y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1, \quad i = 1, 2, ..., r$$

- Wegen verrauschter Daten können die Nebenbedingungen unter Umständen nicht erfüllt werden. Dann gibt es keine Lösung!

Abmilderung der Bedingungen

- Um Fehler in den Daten zu behandeln, werden die Randbedingungen entspannt durch die Einführung von Schlupfvariablen, ξ_i (≥ 0) :

$$\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 1 - \xi_i \qquad \text{for } y_i = 1$$

$$\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \leq -1 + \xi_i \qquad \text{for } y_i = -1.$$

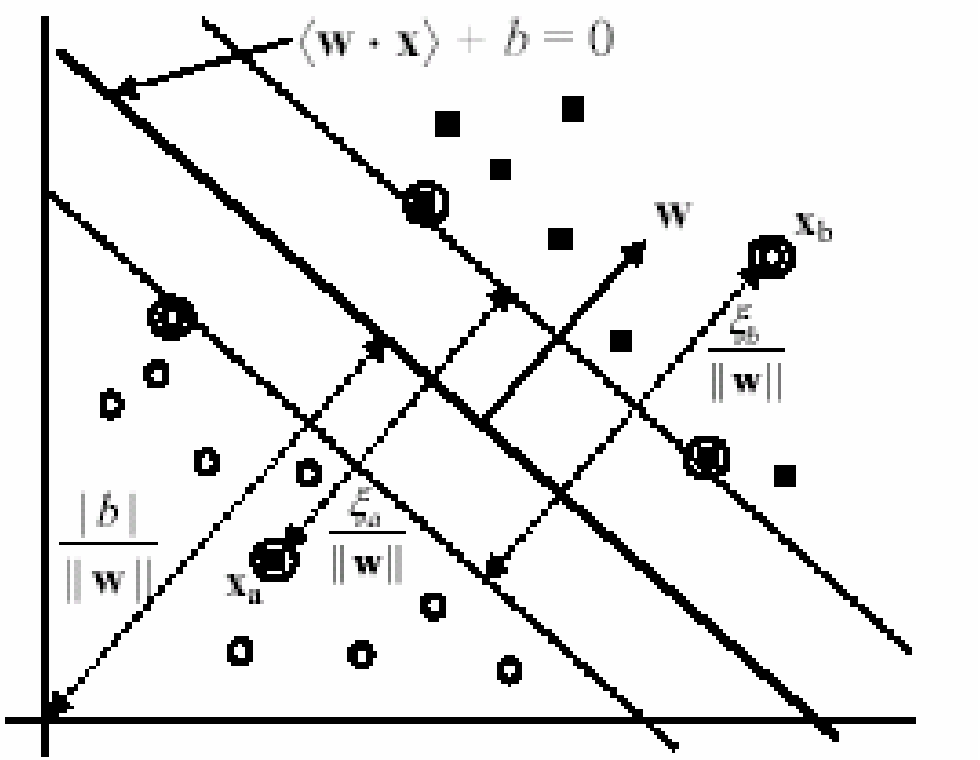
- Neue Nebenbedingungen:

$$y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \; i = 1, \ldots, r,$$

$$\xi_i \geq 0, \;\; i = 1, 2, \ldots, r.$$

Geometrische Interpretation

- Zwei fehlerhafte Datenpunkte $\mathbf{x}_a$ und $\mathbf{x}_b$ (eingekreist) in den falschen Regionen



Bestrafung von Fehlern in der Zielfunktion

- Fehler müssen durch die Zielfunktion bestraft werden.
- Fehler werden mit extra Kosten belastet

$$\text{Minimiere}: \frac{\langle \mathbf{w} \cdot \mathbf{w} \rangle}{2} + C \cdot (\sum_{i=1}^{r} \xi_i)^k \qquad (60)$$

- k = 1 wird oft verwendet, was den Vorteil hat, dass weder ξ_i noch deren Lagrange-Multiplikatoren in der dualen Formulierung auftauchen.

Neues Optimierungsproblem

$$\text{Minimiere}: \quad \frac{\langle \mathbf{w} \cdot \mathbf{w} \rangle}{2} + C \sum_{i=1}^{r} \xi_i \tag{61}$$

$$\text{mit} \quad : \; y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \quad i = 1, 2, ..., r$$

$$\xi_i \geq 0, \quad i = 1, 2, ..., r$$

- Diese Formulierung heißt auch **soft-margin SVM**. Die primäre Lagrange-Funktion ist

(62)

$$L_P = \frac{1}{2}\langle \mathbf{w} \cdot \mathbf{w} \rangle + C \sum_{i=1}^{r} \xi_i - \sum_{i=1}^{r} \alpha_i [y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i] - \sum_{i=1}^{r} \mu_i \xi_i$$

wobei $\alpha_i, \mu_i \geq 0$ die **Lagrange-Multiplikatoren** sind.

Kuhn-Tucker Bedingungen

$$\frac{\partial L_P}{\partial w_j} = w_j - \sum_{i=1}^{r} y_i \alpha_i \mathbf{x}_i = 0, \quad j = 1, 2, \ldots, m \tag{63}$$

$$\frac{\partial L_P}{\partial b} = -\sum_{i=1}^{r} y_i \alpha_i = 0 \tag{64}$$

$$\frac{\partial L_P}{\partial \xi_i} = C - \alpha_i - \mu_i = 0, \quad i = 1, 2, \ldots, r \tag{65}$$

$$y_i((\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i \geq 0, \quad i = 1, 2, \ldots, r \tag{66}$$

$$\xi_i \geq 0, \quad i = 1, 2, \ldots, r \tag{67}$$

$$\alpha_i \geq 0, \quad i = 1, 2, \ldots, r \tag{68}$$

$$\mu_i \geq 0, \quad i = 1, 2, \ldots, r \tag{69}$$

$$\alpha_i(y_i((\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i) = 0, \quad i = 1, 2, \ldots, r \tag{70}$$

$$\mu_i \xi_i = 0, \quad i = 1, 2, \ldots, r \tag{71}$$

Vom Primären zum Dualen

- Wie im linear separierbaren Fall, wird das Primäre Problem in das Duale transformiert, und zwar durch Null-setzen der partiellen Ableitungen der Lagrange-Funktion (62) bezüglich der **primären Variablen** (**w**, *b* und ξ_i), und Rücksubstitution der Ergebnisse Lagrange-Funktion.
  - substituiere Gleichungen (63), (64) und (65) in die primäre Lagrange-Funktion (62).
- Von Gleichung (65), $C - \alpha_i - \mu_i = 0$, kann geschlossen werden, dass $\alpha_i \leq C$ weil $\mu_i \geq 0$.

Duales Problem

- Das duale Problem von (61) ist

$$\text{Maximize: } L_D(\boldsymbol{\alpha}) = \sum_{i=1}^{r} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{r} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle. \qquad (72)$$

$$\text{Subject to: } \sum_{i=1}^{r} y_i \alpha_i = 0$$

$$0 \le \alpha_i \le C, \quad i = 1, 2, \ldots, r.$$

- ξ_i und deren Lagrange-Multiplikatoren μ_i sind nicht in der Gleichung. Die Zielfunktion ist identisch zum separiebaren Fall.
- Der einzige Unterschied ist die Bedingung $\alpha_i \le C$.

Werte für primäre Variablen

- Das duale Problem (72) kann numerisch gelößt werden.
- Die gefundenen α_i Werte werden genutzt um **w** und *b* zu berechnen. **w** ergibt sich nach Gleichung (63) und *b* nach den Kuhn-Tucker Bedingungen (70) und (71).
- Problem: keine Werte für ξ_i berechnet
  - Aus den Gleichungen (65), (70) und (71) sieht man, dass wenn $0 < \alpha_i < C$ dann sind $\xi_i = 0$ und $y_i\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b - 1 + \xi_i = 0$. D.h. jeder Trainingsdatenpunkt mit $0 < \alpha_i < C$ kann genutzt werden um aus Gleichung (69) (mit $\xi_i = 0$) *b* zu berechnen.

$$b = \frac{1}{y_i} - \sum_{i=1}^{r} y_i \alpha_i \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle = 0. \qquad (73)$$

(65), (70) und (71) sagen noch mehr aus

$$\begin{aligned} \alpha_i = 0 \quad &\Rightarrow \quad y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1 \text{ and } \xi_i = 0 \\ 0 < \alpha_i < C \quad &\Rightarrow \quad y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) = 1 \text{ and } \xi_i = 0 \\ \alpha_i = C \quad &\Rightarrow \quad y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \leq 1 \text{ and } \xi_i \geq 0 \end{aligned} \tag{74}$$

- (74) zeigt wichtige Eigenschaft von SVM.
  - Lösung ist **dünn besetzt** in α_i. Viele Trainingsdatenpunkte sind außerhalb des Randes ihre α_i's in der Lösung sind 0.
  - Nur solche Datenpunkte die auf den Rand ($y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) = 1$, Support-Vektoren im separierbaren Fall), innerhalb des Randes ($\alpha_i = C$ und $y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) < 1$), oder Fehler sind nicht-null.
  - Ohne dünn besetzte Lösung wäre SVM nicht anwendbar für große Datenmengen.

Entscheidungsgrenze

- Die Entscheidungsgrenze ist (beachte dass viele α_i's 0 sind)

$$\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = \sum_{i=1}^{r} y_i \alpha_i \langle \mathbf{x}_i \cdot \mathbf{x} \rangle + b = 0 \qquad (75)$$

- Die Entscheidungsregel für Testbeispiele ist die gleiche wie im separierbaren Fall,

$$sign(\langle \mathbf{w} \cdot \mathbf{x} \rangle + b).$$

- Der Parameter *C* in der Zielfunktion wird meist durch eine Validierungsmenge oder Kreuz-Validierung bestimmt.

Nicht-lineare Separierung?

- SVM erfordern lineare Separierung.
- Reale Datenmenge können nicht-lineare Separierung erfordern.
- Um nicht-lineare Separierung zu ermöglichen, können die gleiche Theorie und Techniken wie im linearen Fall genutzt werden.
- Nur die Eingabedaten werden in einen anderen Datenraum transformiert, der meist eine viel höhere Dimensionalität hat, s.d.
  - eine lineare Entscheidungsgrenze die positiven und negativen Beispiel im transformierten Raum separieren kann,
- Der transformierte Raum heißt **Eigenschaftsraum**. Der Originalraum heißt **Eingaberaum**.

Transformation

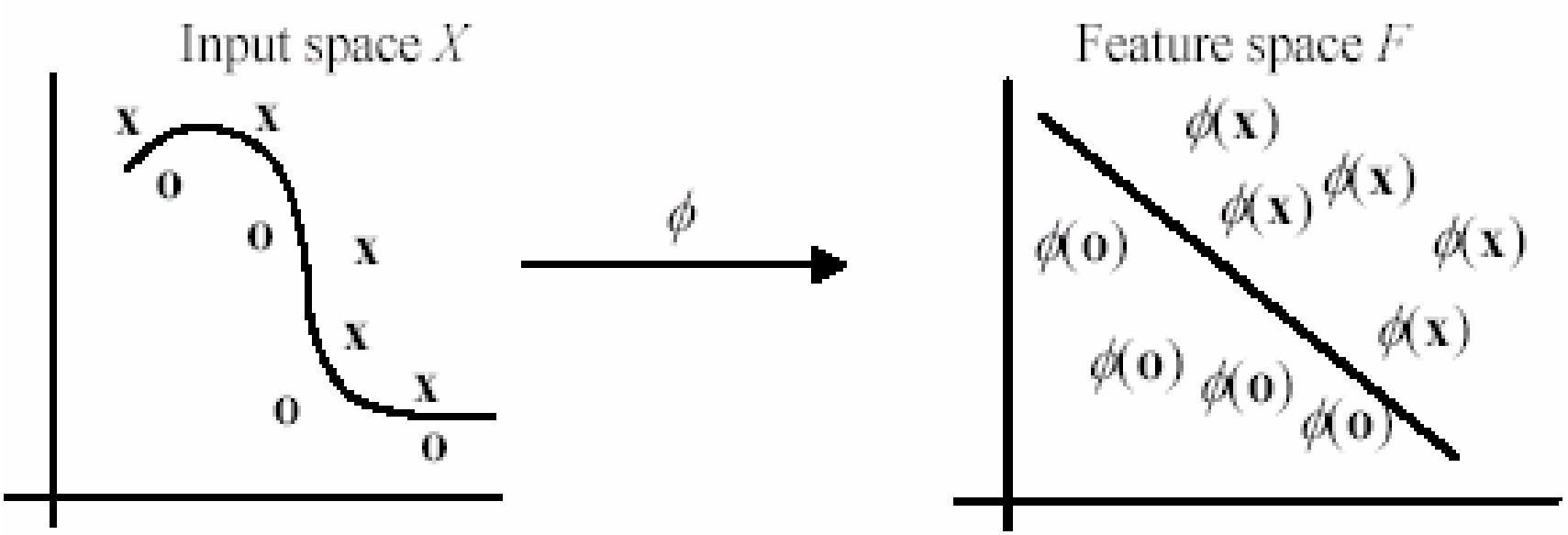
- Die Grundidee ist die Daten aus dem Eingaberaum X in den Eigenschaftsraum F nicht-linear abzubilden durch eine Funktion ϕ,

$$\begin{aligned} \phi &: X \to F \\ \mathbf{x} &\mapsto \phi(\mathbf{x}) \end{aligned} \qquad (76)$$

- Nach der Abbildung werden die Original-trainingsdaten $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_r, y_r)\}$ zu:

$$\{(\phi(\mathbf{x}_1), y_1), (\phi(\mathbf{x}_2), y_2), \ldots, (\phi(\mathbf{x}_r), y_r)\} \qquad (77)$$

Geometrische Interpretation



- In diesem Beispiel ist der transformierte Raum auch 2-D. Meist ist aber die Anzahl der Dimensionen des Eigenschaftsraumes viel höher als die des Eingaberaumes.

Optimierungs Problem in (61)

With the transformation, the optimization problem in (61) becomes

$$\text{Minimize}: \quad \frac{\langle \mathbf{w} \cdot \mathbf{w} \rangle}{2} + C \sum_{i=1}^{r} \xi_i \tag{78}$$

$$\text{Subject to}: \quad y_i(\langle \mathbf{w} \cdot \phi(\mathbf{x}_i) \rangle + b) \geq 1 - \xi_i, \quad i = 1, 2, \ldots, r$$

$$\xi_i \geq 0, \quad i = 1, 2, \ldots, r$$

The dual is

$$\text{Maximize}: \quad L_D = \sum_{i=1}^{r} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{r} y_i y_j \alpha_i \alpha_j \langle \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) \rangle. \tag{79}$$

$$\text{Subject to}: \quad \sum_{i=1}^{r} y_i \alpha_i = 0$$

$$0 \leq \alpha_i \leq C, \quad i = 1, 2, \ldots, r.$$

The final decision rule for classification (testing) is

$$\sum_{i=1}^{r} y_i \alpha_i \langle \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}) \rangle + b \tag{80}$$

Eine Beispiel-Transformation

- Angenommen der Eingaberaum ist 2-dimensional, folgende Transformation bildet vom 2-D in den 3-D ab:

$$(x_1, x_2) \mapsto ({x_1}^2, {x_2}^2, \sqrt{2} x_1 x_2)$$

- Das Trainingsbeispiel ((2, 3), -1) im Eingaberaum wird in den Eigenschaftsraum transformiert:

  ((4, 9, 8.5), -1)

Problem expliziter Transformation

- Explizite Datentransformation und nachfolgende Anwendung einer linearen SVM hat mehrere Probleme.
  - Die Anzahl der Dimensionen im Eigenschaftsraum ist sehr groß bei einigen nutzlichen Transformationen, auch wenn die Anzahl der Attribute im Eingaberaum klein ist.
  - Erzeugt dadurch enormen Resourcenverbrauch.
- Explizite Transformation wird jedoch nicht gebraucht.

Kern-Funktionen

- In der dualen Formulierung brauchen
  - die Konstruktion der optimale Hyperebene (79) in *F* und
  - die Evaluation der zugehörigen Entscheidungsfunktion (80)

  nur Skalarprodukte $\langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle$ und nie die transformierten Vektor $\phi(\mathbf{x})$ in seiner expliziten Form. Das ist sehr wichtig.
- D.h., falls das Skalarprodukt $\langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle$ über die Eingabevektoren **x** und **z** direkt berechnet werden kann,
  - braucht man die Eigenschaftsvektoren $\phi(\mathbf{x})$ oder ϕ selbst nicht.
- In SVM wird dies durch Kernfunktionen getan,

$$K(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle \qquad (82)$$

Eine Beispiel Kernfunktion

- Polynomieller Kern

$$K(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x} \cdot \mathbf{z} \rangle^d \tag{83}$$

- Der Kern vom Grad $d = 2$ in einem 2-dimensionalen Raum: $\mathbf{x} = (x_1, x_2)$ und $\mathbf{z} = (z_1, z_2)$.

$$\begin{aligned}
\langle \mathbf{x} \cdot \mathbf{z} \rangle^2 &= (x_1 z_1 + x_2 z_2)^2 \\
&= x_1^2 z_1^2 + 2 x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \qquad (84) \\
&= \langle (x_1^2, x_2^2, \sqrt{2} x_1 x_2) \cdot (z_1^2, z_2^2, \sqrt{2} z_1 z_2) \rangle \\
&= \langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle,
\end{aligned}$$

- Dies zeigt das der Kern $\langle \mathbf{x} \cdot \mathbf{z} \rangle^2$ ein Skalarprodukt im transformierten Raum ist

Kernel-Trick

- Die Herleitung in (84) ist nur zur Illustration.
- Die Abbildung braucht nicht expliziet gefunden zu werden.
- Es kann einfach die Kernfunktion direkt angewendet werden
  - ersetze alle Skalarprodukte $\langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle$ in (79) und (80) mit der Kernfunktion $K(\mathbf{x}, \mathbf{z})$ (z.B., dem polynomiellen Kern $\langle \mathbf{x} \cdot \mathbf{z} \rangle^d$ in (83)).
- Diese Strategie heißt **Kernel-Trick**.

Ist es eine Kernfunktion?

- Die Frage ist: Wie weiß man, ob eine Funktion ein Kern ist, ohne die Herleitung wir in (84) durchzuführen? d.h.,
  - Wie weiß man dass eine Kernfunktion tatsächlich ein Skalarprodukt in einem Eigenschaftsraum ist?
- Diese Frage wird durch Mercer's Theorem, beantwortet, welches aber nicht hier vorgestellt wird.

Häufig genutzte Kerne

- Die Idee der Kerne verallgemeinert das Skalarprodukt im Eingaberaum. Dieses Skalarprodukt ist auch ein Kern, mit der Identität als Abbildung

$$K(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x} \cdot \mathbf{z} \rangle. \tag{85}$$

Commonly used kernels include

$$\text{Polynomial:} \quad K(\mathbf{x}, \mathbf{z}) = (\langle \mathbf{x} \cdot \mathbf{z} \rangle + \theta)^d \tag{86}$$

$$\text{Gaussian RBF:} \quad K(\mathbf{x}, \mathbf{z}) = e^{-\|\mathbf{x}-\mathbf{z}\|^2 / 2\sigma} \tag{87}$$

$$\text{Sigmoidal:} \quad K(\mathbf{x}, \mathbf{z}) = \tanh(k\langle \mathbf{x} \cdot \mathbf{z} \rangle - \delta) \tag{88}$$

where $\theta \in R$, $d \in N$, $\sigma > 0$, and $k, \delta \in R$.

Beschränkungen von SVM

- SVM funktionieren nur im kontinierlichen Raum. Kategorische Attribute müssen in numerische Werte konvertiert werden.
- SVM können nur Zweiklassen-Klassifikation. Für Mehr-Klassen-Probleme müssen Strategien wie Eine-Gegen-den-Rest angewendet werden.
- Die gefundene Hyperebene is schwer für Anwender zu verstehen. Kerne machen das nur komplizierter. Deshalb werden SVM meist dann eingesetzt wenn die Anwendung es nicht erfordert, dass ein Mensch die Entscheidung versteht.

Road Map

- Basic concepts
- Decision tree induction
- Evaluation of classifiers
- Rule induction
- Classification using association rules
- Naïve Bayesian classification
- Naïve Bayes for text classification
- Support vector machines
- **K-nearest neighbor**
- Ensemble methods: Bagging and Boosting
- Summary

k-Nearest Neighbor Classification (kNN)

- Unlike all the previous learning methods, kNN does not build model from the training data.
- To classify a test instance *d*, define *k*-neighborhood *P* as *k* nearest neighbors of *d*
- Count number *n* of training instances in *P* that belong to class c_j
- Estimate $\Pr(c_j|d)$ as n/k
- No training is needed. Classification time is linear in training set size for each test case.

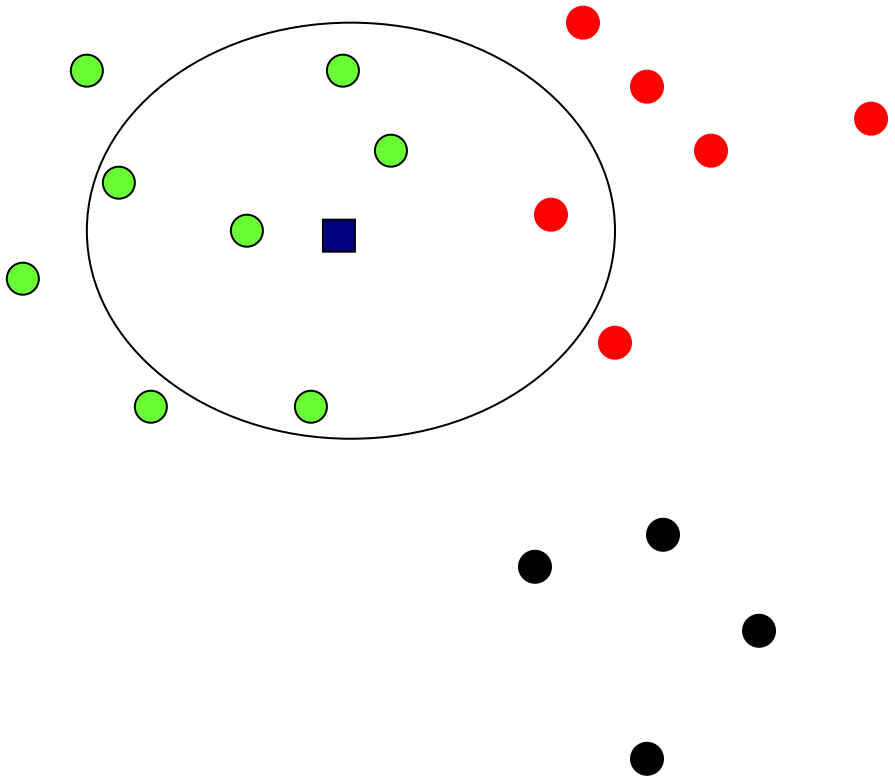
kNNAlgorithm

Algorithm kNN(D, d, k)

1 Compute the distance between d and every example in D;
2 Choose the k examples in D that are nearest to d, denote the set by P ($\subseteq D$);
3 Assign d the class that is the most frequent class in P (or the majority class);

- k is usually chosen empirically via a validation set or cross-validation by trying a range of k values.
- Distance function is crucial, but depends on applications.

Example: k=6 (6NN)



Government

Science

Arts

A new point ■

Pr(science| ■)?

Discussions

- kNN can deal with complex and arbitrary decision boundaries.
- Despite its simplicity, researchers have shown that the classification accuracy of kNN can be quite strong and in many cases as accurate as those elaborated methods.
- kNN is slow at the classification time
- kNN does not produce an understandable model

Road Map

- Basic concepts
- Decision tree induction
- Evaluation of classifiers
- Rule induction
- Classification using association rules
- Naïve Bayesian classification
- Naïve Bayes for text classification
- Support vector machines
- K-nearest neighbor
- **Ensemble methods: Bagging and Boosting**
- Summary

Combining classifiers

- So far, we have only discussed individual classifiers, i.e., how to build them and use them.
- Can we combine multiple classifiers to produce a better classifier?
- Yes, sometimes
- We discuss two main algorithms:
  - Bagging
  - Boosting

Bagging

- Breiman, 1996
- Bootstrap Aggregating = Bagging
  - Application of bootstrap sampling
    - Given: set *D* containing *m* training examples
    - Create a sample *S*[*i*] of *D* by drawing *m* examples at random *with replacement* from *D*
    - *S*[*i*] of size *m*: expected to leave out 0.37 of examples from *D*

Bagging (cont…)

- Training
  - Create *k* bootstrap samples *S*[1], *S*[2], …, *S*[*k*]
  - Build a distinct classifier on each *S*[*i*] to produce *k* classifiers, using the same learning algorithm.
- Testing
  - Classify each new instance by voting of the *k* classifiers (equal weights)

Bagging Example

Original	1	2	3	4	5	6	7	8
Training set 1	2	7	8	3	7	6	3	1
Training set 2	7	8	5	6	4	2	7	1
Training set 3	3	6	2	7	5	6	2	2
Training set 4	4	5	1	4	6	4	3	8

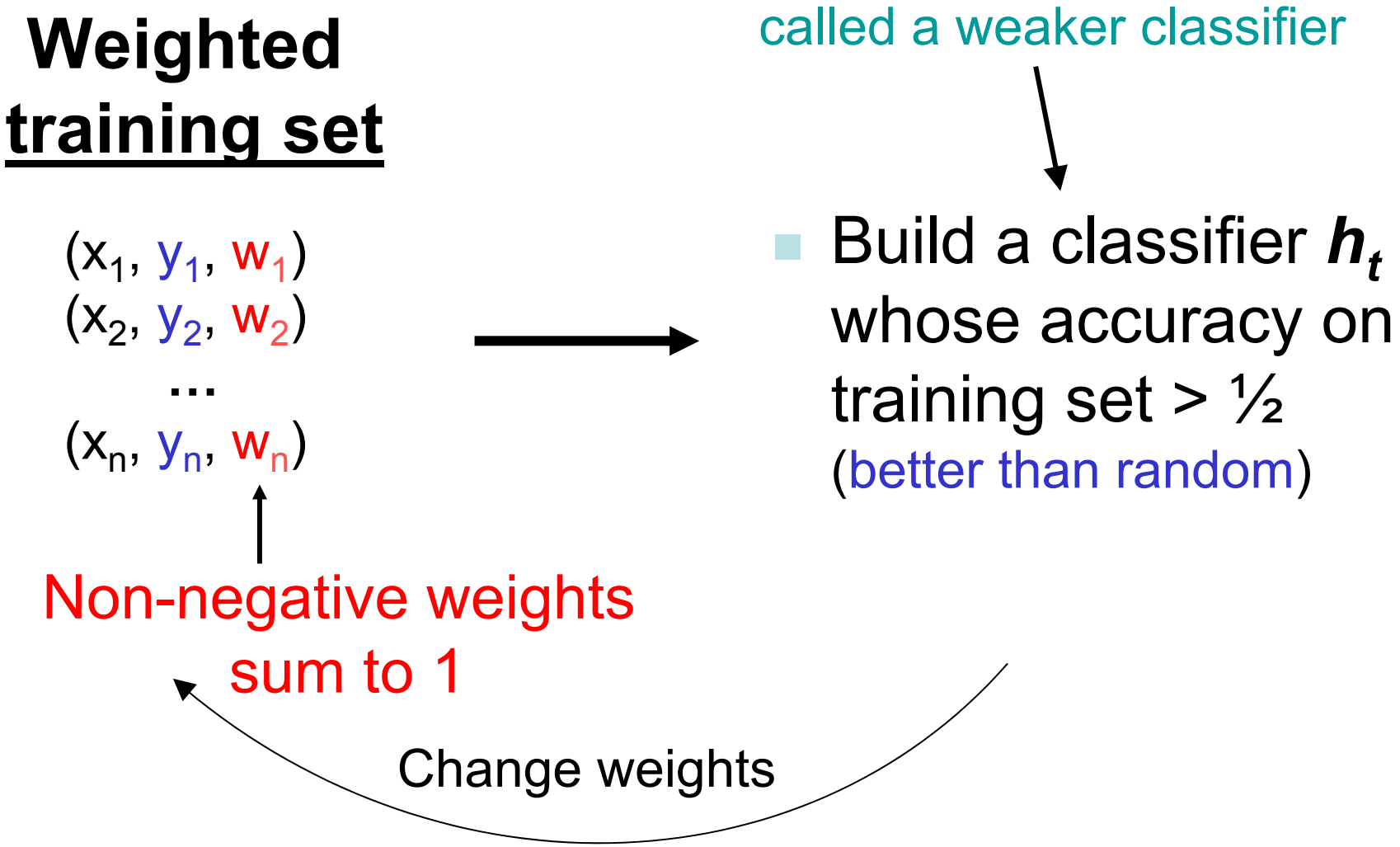
Bagging (cont …)

- When does it help?
  - When learner is unstable
    - Small change to training set causes large change in the output classifier
    - True for decision trees, neural networks; not true for *k*-nearest neighbor, naïve Bayesian, class association rules
  - Experimentally, bagging can help substantially for unstable learners, may somewhat degrade results for stable learners

Boosting

- A family of methods:
  - We only study **AdaBoost** (Freund & Schapire, 1996)
- Training
  - Produce a sequence of classifiers (the same base learner)
  - Each classifier is dependent on the previous one, and focuses on the previous one's errors
  - Examples that are incorrectly predicted in previous classifiers are given higher weights
- Testing
  - For a test case, the results of the series of classifiers are combined to determine the final class of the test case.

AdaBoost


Weighted training set
(x_1, y_1, w_1)
(x_2, y_2, w_2)
...
(x_n, y_n, w_n)
called a weaker classifier
Build a classifier h_t whose accuracy on training set > ½ (better than random)
Non-negative weights sum to 1
Change weights

AdaBoost algorithm

Algorithm AdaBoost.M1

Input: sequence of m examples $\langle (x_1, y_1), \ldots, (x_m, y_m) \rangle$
with labels $y_i \in Y = \{1, \ldots, k\}$
weak learning algorithm **WeakLearn**
integer T specifying number of iterations

Initialize $D_1(i) = 1/m$ for all i.

Do for $t = 1, 2, \ldots, T$:

1. Call **WeakLearn**, providing it with the distribution D_t.
2. Get back a hypothesis $h_t : X \to Y$.
3. Calculate the error of h_t: $\epsilon_t = \sum_{i: h_t(x_i) \neq y_i} D_t(i)$.

   If $\epsilon_t > 1/2$, then set $T = t - 1$ and abort loop.
4. Set $\beta_t = \epsilon_t / (1 - \epsilon_t)$.
5. Update distribution D_t:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & \text{if } h_t(x_i) = y_i \\ 1 & \text{otherwise} \end{cases}$$

   where Z_t is a normalization constant (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$h_{fin}(x) = \arg\max_{y \in Y} \sum_{t: h_t(x) = y} \log \frac{1}{\beta_t}.$$

Bagging, Boosting and C4.5

C4.5's mean error rate over the 10 cross-validation.

Bagged C4.5 vs. C4.5.

Boosted C4.5 vs. C4.5.

Boosting vs. Bagging

	C4.5	Bagged C4.5 vs C4.5			Boosted C4.5 vs C4.5			Boosting vs Bagging	
	err (%)	err (%)	w-l	ratio	err (%)	w-l	ratio	w-l	ratio
anneal	7.67	6.25	10-0	.814	4.73	10-0	.617	10-0	.758
audiology	22.12	19.29	9-0	.872	15.71	10-0	.710	10-0	.814
auto	17.66	19.66	2-8	1.113	15.22	9-1	.862	9-1	.774
breast-w	5.28	4.23	9-0	.802	4.09	9-0	.775	7-2	.966
chess	8.55	8.33	6-2	.975	4.59	10-0	.537	10-0	.551
colic	14.92	15.19	0-6	1.018	18.83	0-10	1.262	0-10	1.240
credit-a	14.70	14.13	8-2	.962	15.64	1-9	1.064	0-10	1.107
credit-g	28.44	25.81	10-0	.908	29.14	2-8	1.025	0-10	1.129
diabetes	25.39	23.63	9-1	.931	28.18	0-10	1.110	0-10	1.192
glass	32.48	27.01	10-0	.832	23.55	10-0	.725	9-1	.872
heart-c	22.94	21.52	7-2	.938	21.39	8-0	.932	5-4	.994
heart-h	21.53	20.31	8-1	.943	21.05	5-4	.978	3-6	1.037
hepatitis	20.39	18.52	9-0	.908	17.68	10-0	.867	6-1	.955
hypo	.48	.45	7-2	.928	.36	9-1	.746	9-1	.804
iris	4.80	5.13	2-6	1.069	6.53	0-10	1.361	0-8	1.273
labor	19.12	14.39	10-0	.752	13.86	9-1	.725	5-3	.963
letter	11.99	7.51	10-0	.626	4.66	10-0	.389	10-0	.621
lymphography	21.69	20.41	8-2	.941	17.43	10-0	.804	10-0	.854
phoneme	19.44	18.73	10-0	.964	16.36	10-0	.842	10-0	.873
segment	3.21	2.74	9-1	.853	1.87	10-0	.583	10-0	.684
sick	1.34	1.22	7-1	.907	1.05	10-0	.781	9-1	.861
sonar	25.62	23.80	7-1	.929	19.62	10-0	.766	10-0	.824
soybean	7.73	7.58	6-3	.981	7.16	8-2	.926	8-1	.944
splice	5.91	5.58	9-1	.943	5.43	9-0	.919	6-4	.974
vehicle	27.09	25.54	10-0	.943	22.72	10-0	.839	10-0	.889
vote	5.06	4.37	9-0	.864	5.29	3-6	1.046	1-9	1.211
waveform	27.33	19.77	10-0	.723	18.53	10-0	.678	8-2	.938
average	*15.66*	*14.11*		*.905*	*13.36*		*.847*		*.930*

Does AdaBoost always work?

- The actual performance of boosting depends on the data and the base learner.
  - It requires the base learner to be unstable as bagging.
- Boosting seems to be susceptible to noise.
  - When the number of outliners is very large, the emphasis placed on the hard examples can hurt the performance.

Road Map

- Basic concepts
- Decision tree induction
- Evaluation of classifiers
- Rule induction
- Classification using association rules
- Naïve Bayesian classification
- Naïve Bayes for text classification
- Support vector machines
- K-nearest neighbor
- **Summary**

Summary

- Applications of supervised learning are in almost any field or domain.
- We studied 8 classification techniques.
- There are still many other methods, e.g.,
  - Bayesian networks
  - Neural networks
  - Genetic algorithms
  - Fuzzy classification

  This large number of methods also show the importance of classification and its wide applicability.
- It remains to be an active research area.