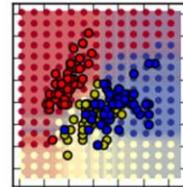
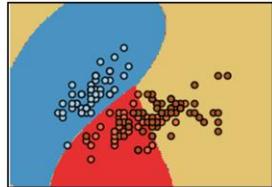




Bildanalyse und Visualisierung in Diagnostik und Therapie Support Vector Machines & Random Decision Forests



M.Sc. Oskar Maier

Prof. Dr. rer. nat. habil. Heinz Handels

Institut für Medizinische Informatik

Universität zu Lübeck

Da die Folien alleine nicht alle Informationen wiedergeben, habe ich zu jeder ein paar Notizen hinzugefügt, die das Lernen bzw. Nachschlagen erleichtern sollten.

Viel Spaß damit,

Oskar



Inhalt

- Klassifizierung
- Support Vector Machines (SVM)
 - Maximum-Margin Klassifizierer
 - Nicht-lineare Probleme
 - Kernel-Trick
 - Soft-Margin Klassifizierer
- Entscheidungsbäume
 - Training und Anwendung
 - Informationsgewinn als Gütemaß
- Random Decision Forests (RDF)
 - Mehr ist besser
 - Das Random in Random Decision Forests
- Anwendungsbeispiele



Klassifizierung



Definitionen

- **Klassifizierung:** „Die Einordnung einer neuen Observation in eine von mehreren Kategorien anhand ihrer Merkmale“
- **Merkmale:** individuelle, messbare Eigenschaften einer Observation
 - Unterscheidend
 - Unabhängig
- **Klassifizierer:** Ein auf einer Trainingsmenge erlerntes Problemlösungsverfahren
- **Trainingsmenge:** Eine Menge von Stichproben-Observationen mit bekannten Klassenzugehörigkeiten zum Trainieren eines Klassifizierers
- Im Maschinellen Lernen: **supervised training**



Ein Beispiel

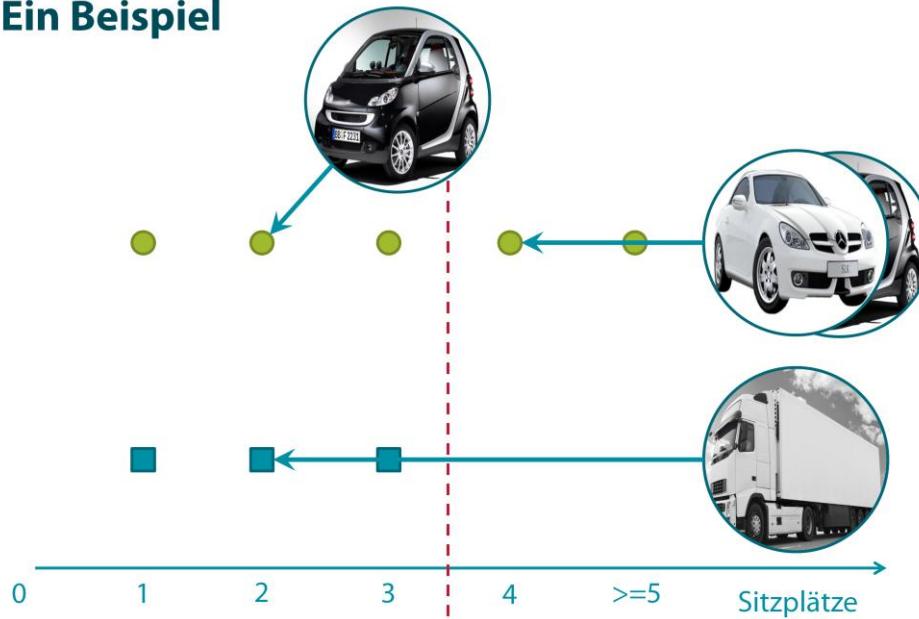


LKW

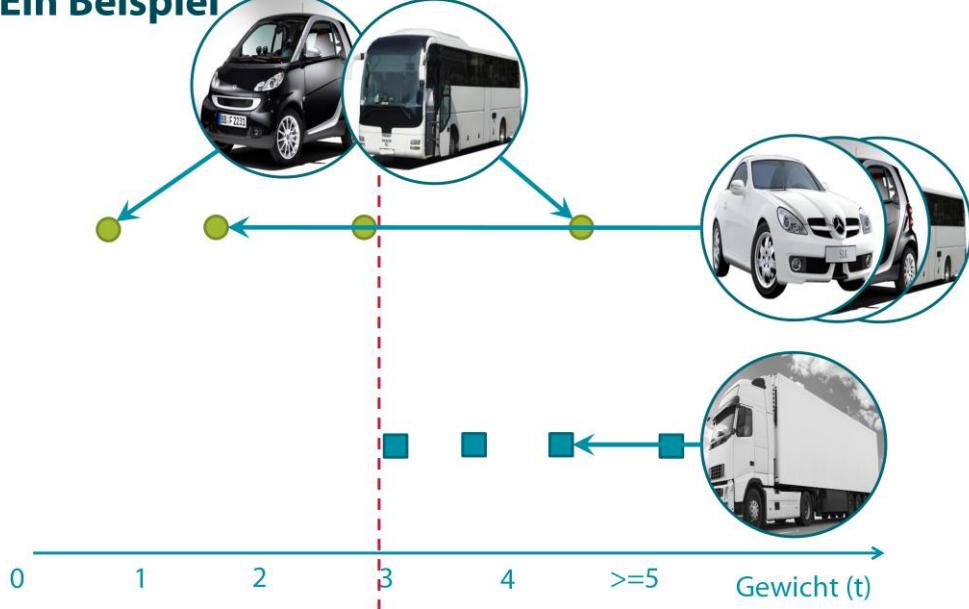


Andere
Kraftfahrzeuge

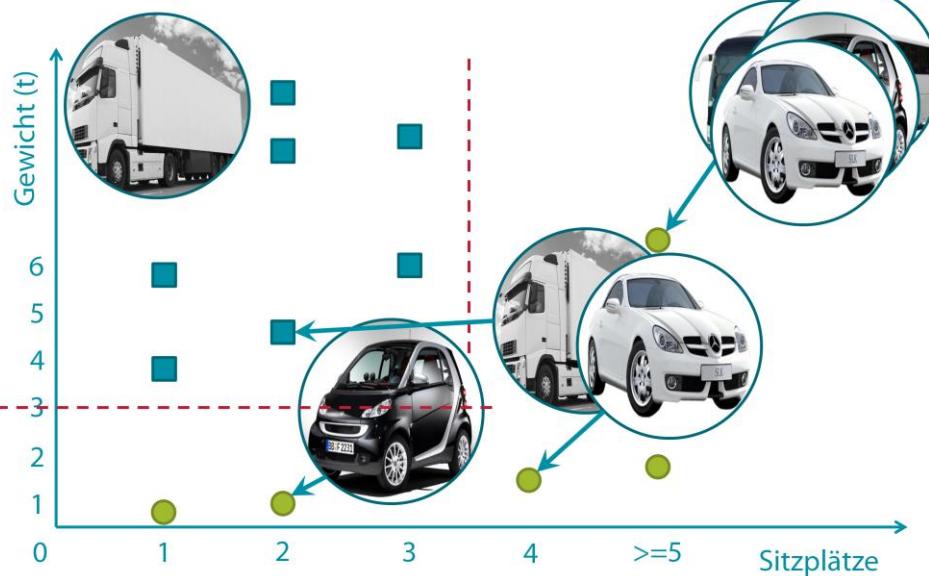
Ein Beispiel



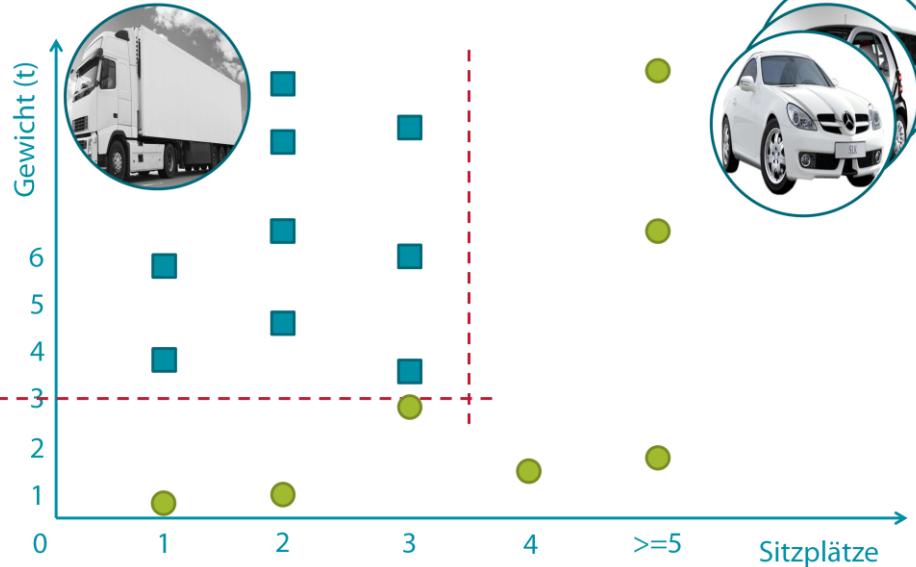
Ein Beispiel



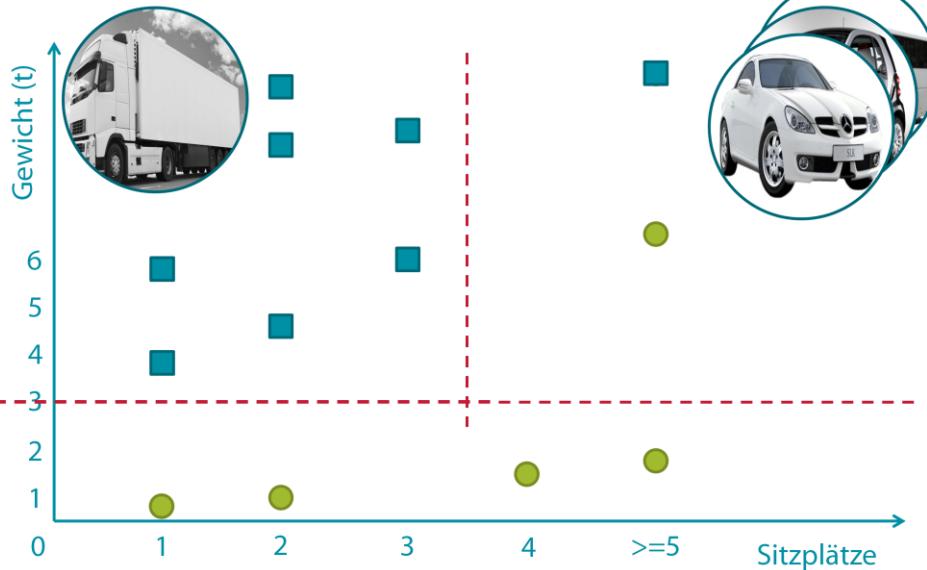
Ein Beispiel



Ein Beispiel - Anwendung



Ein Beispiel – Trainingsset



10



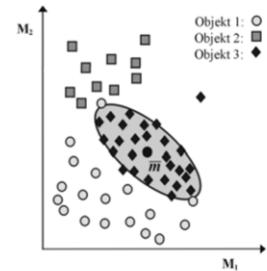
Formales Vorgehen

1. Klassifizierungsproblem bestimmen
2. Menge von Stichproben-Observationen mit zugehöriger Klasse sammeln => Trainingsmenge
 - Repräsentativ?
 - Verrauscht?
 - Nicht zu wenig (repräsentieren Modell nicht)
 - Evaluationsmenge zu Seite legen
3. Geeignete Merkmale extrahieren
 - Unterscheidend und unabhängig?
 - Nicht zu viele (Fluch der Dimensionalität)
 - Nicht zu wenig (keine Abgrenzung möglich)
4. Geeigneten Klassifizierer auswählen
 - Wie sieht der Problemraum aus?
5. Klassifizierer anlernen
6. Genauigkeit und Generalisierung des Klassifizierers evaluieren

Rückblick

„Algorithmus zur ROI-basierten Pixelklassifikation in multi-spektralen Bilddaten“, 2te Vorlesung

1. Auswahl der ROI => Trainingsdaten
2. Modell lernen => Klassifizierer anlernen
(Stichwort: Mahalanobis-Distanz / CovM)
3. Verbleibende Voxel klassifizieren =>
Klassifizierer anwenden





Unsupervised learning

- Stichwort: [k-means clustering](#)
- Beim unsupervised learning sind die Klassenzugehörigkeiten (kurz: Klassen) nicht bekannt
- Zusammengehörigkeit wird über die (meist euklidische) Nähe der Merkmale im Merkmalsraum definiert: Mustererkennung



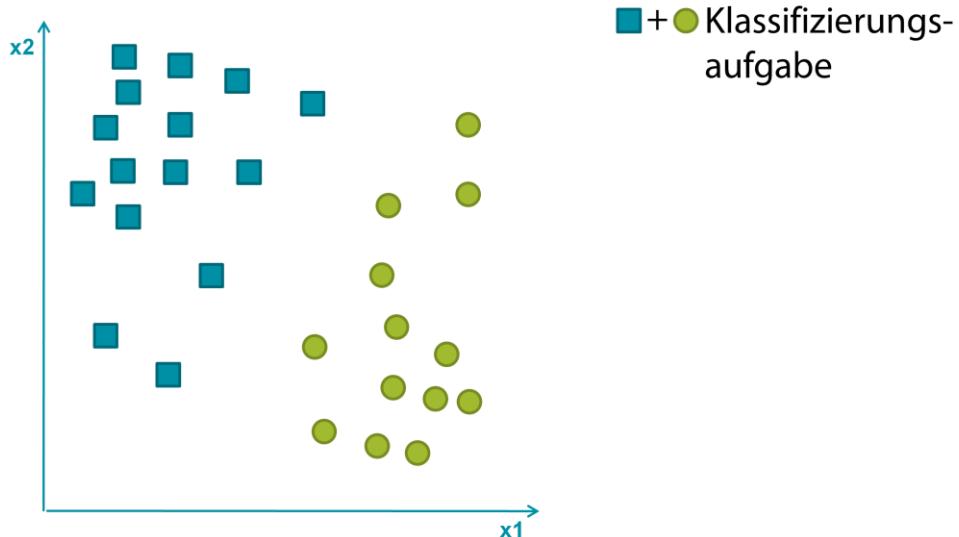
Klassifizierung

Q&A



Support Vector Machines (SVM)

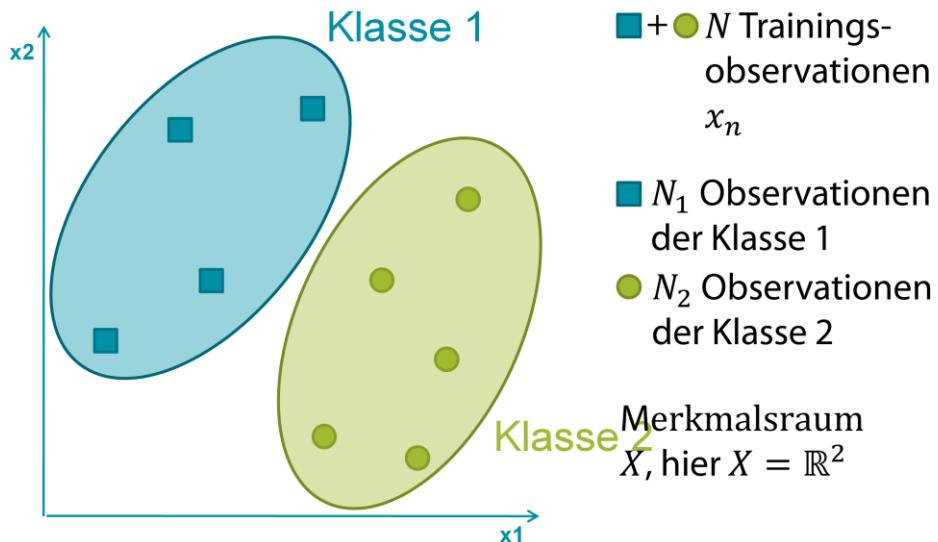
Support Vector Machines (SVM)



16

- Aufgabe des Klassifikators ist zwischen den beiden Mengen zu unterscheiden zu lernen

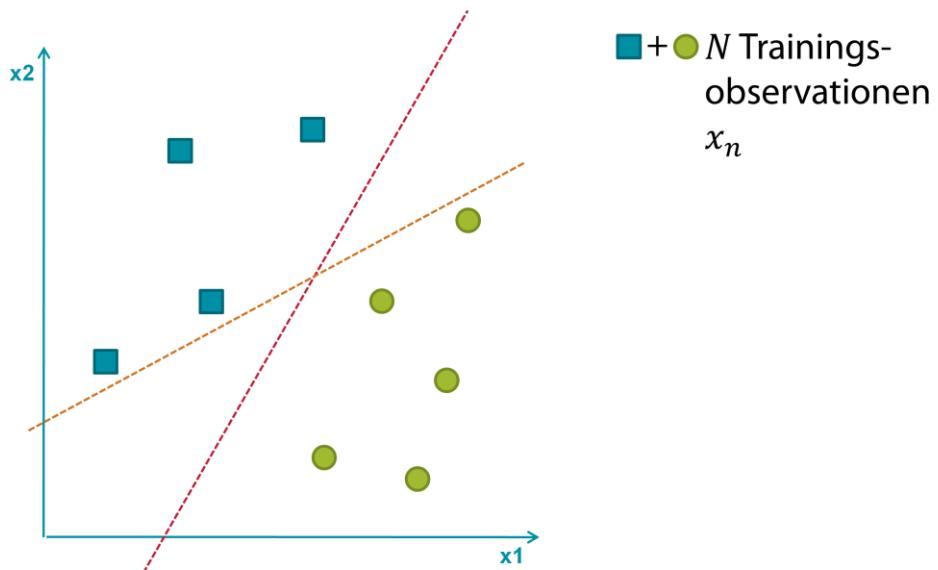
Support Vector Machines (SVM)



17

- Training erfolgt anhand von einigen wenigen (meist zufällig gezogenen) Trainingsobservationen
- Klassifikator soll lernen zu Generalisieren (Verallgemeinern)
- Jedes Element wird beschrieben durch zwei reelle Wert d.h. $x=(x_1,x_2)$

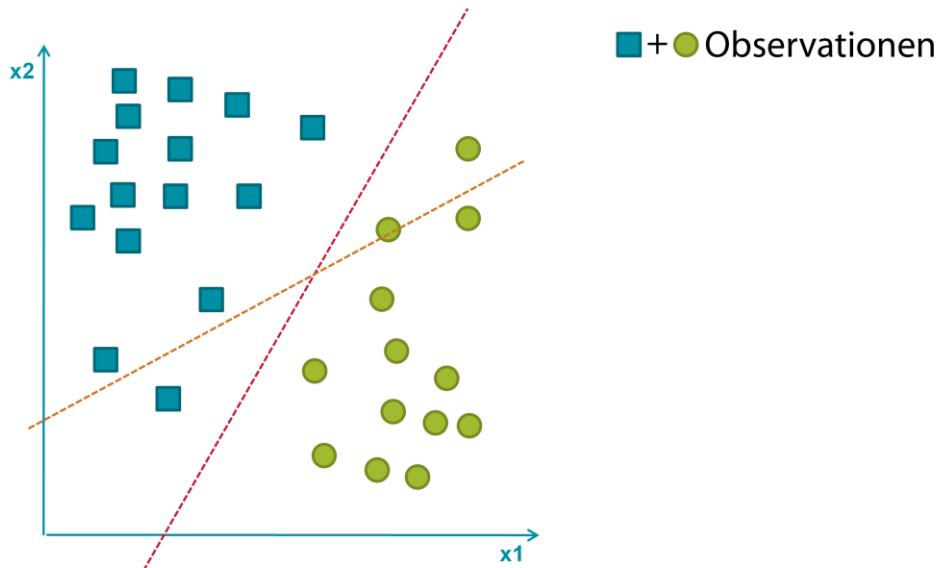
SVM: Partitionieren der Observationen



18

- Unendlich Linien separieren die Trainingsobservationen korrekt
- Aber welche ist die Beste?

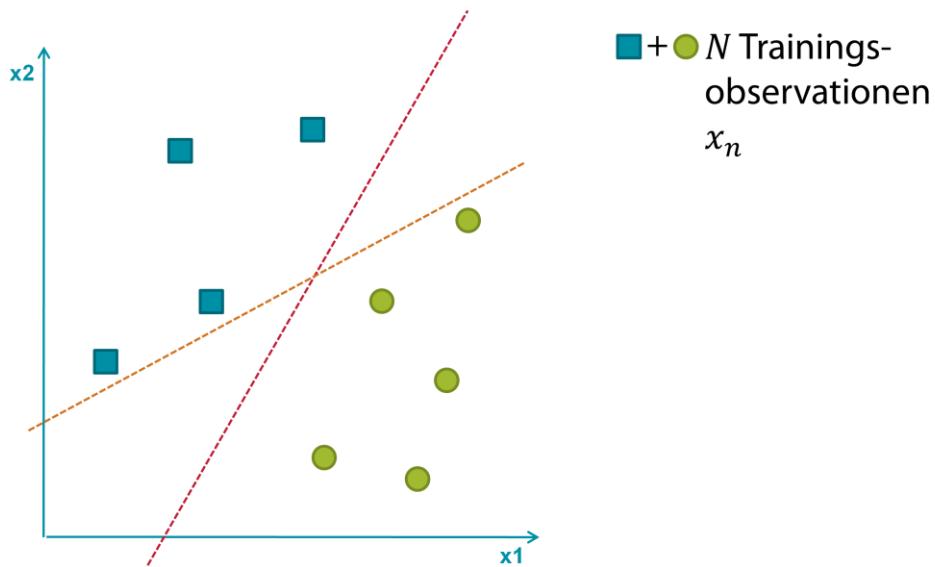
SVM: Partitionieren der Observationen



19

- Betrachtet man die Klassifizierungsaufgabe komplett, wird es klar
- Die rote Trennlinie **generalisiert** besser

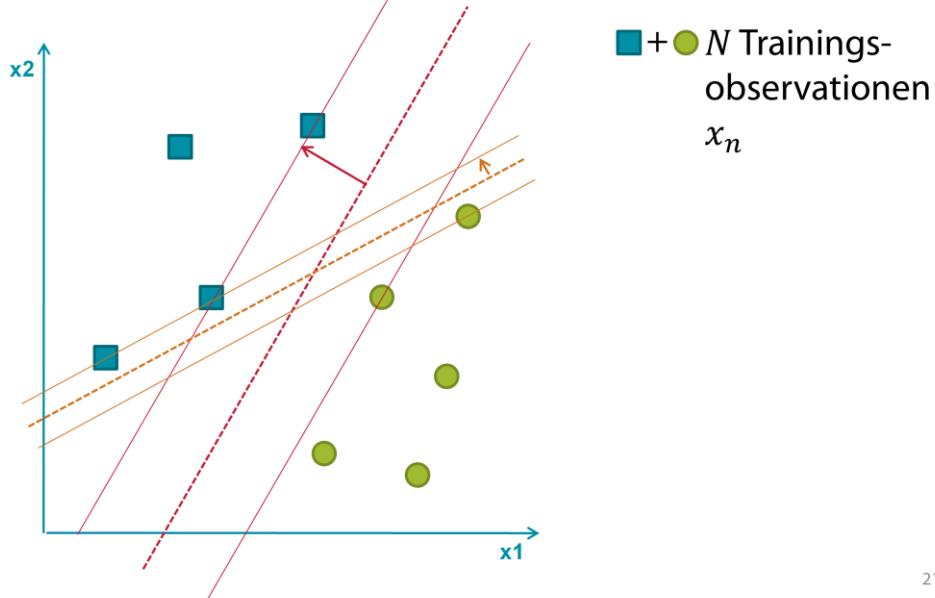
SVM: Partitionieren der Observationen



20

- Doch wie kann man die zu erwartende Güte einer Linie nur Anhand der Trainingsobservationen bestimmen?

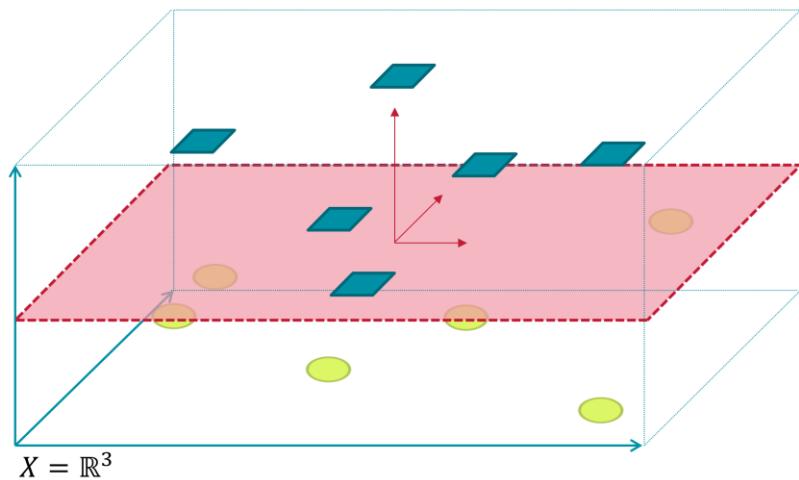
SVM: Maximum Margin Klassifikator



21

- Über den Margin d.h. Abstand zur nächsten Trainingsobservation egal welcher Menge.
- Wird dieser maximiert kann eine maximale Generalisierung und damit ein minimaler zu erwartender Fehler angenommen werden.
- Warum? Man geht auf Nummer sicher, da die Trainingsobservationen nicht jeden Fall abdecken können, jedoch als repräsentativ angenommen werden.

SVM: Die partitionierende Hyperebene



22

- Wenn der Observationsraum größer ist (z.B. \mathbb{R}^3 hier), dann wird aus der partitionierenden Linie / Trennlinie eine partitionierende Hyperebene
- Diese Unterteilungen / Klassifizierung ist immer linear d.h. die partitionierende Hyperebene hat nie eine gekrümmte Oberfläche (werden wir später drauf zurück kommen)



SVM

Wir suchen die Hyperebene, die

1. die Trainingsdaten perfekt trennt und
2. den maximalen Abstand zu den nächsten Trainingsobservationen beider Klassen aufweist

Wir benötigen für eine rechnerische Lösung:

- Mathematische Beschreibung der Hyperebene
- Mathematische Beschreibung des Abstandes
- Mathematische Beschreibung des Optimierungsproblems

SVM: Die partitionierende Hyperebene

$$\mathbf{w}^T \mathbf{x} + b = 0 \quad b \in \mathbb{R}$$

(Normalenform)

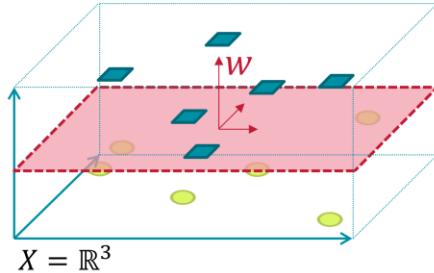
\sim Schwellenwert

$\mathbf{w} \in \mathcal{X}$

Ausrichtung der Ebene
Skalierung beliebig

\mathbf{w}, b

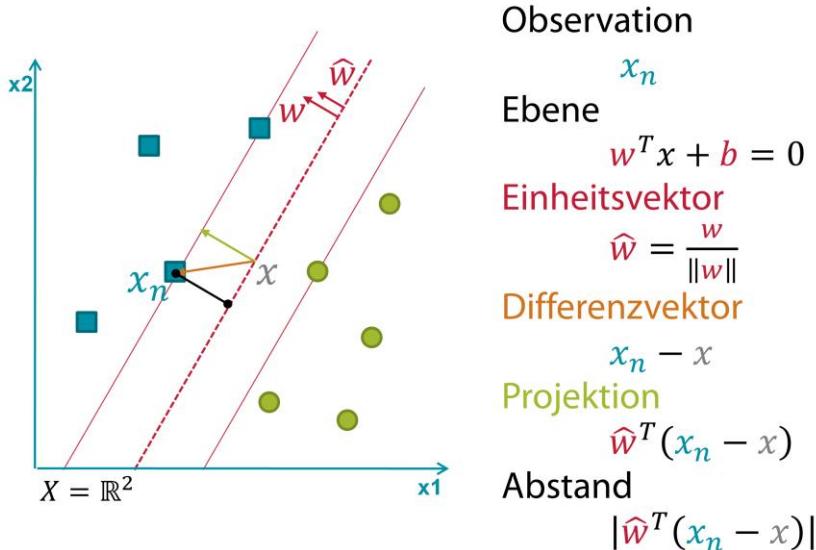
Unbekannte



24

- Wir suchen also eine partitionierende Hyperebene. Doch wie das Problem formulieren, um es algorithmisch lösen zu können?
- Erste einmal die Ebene: Sie wird durch eine Form bestimmt, die der Normalenform sehr ähnlich ist.
- Der Parameter b entspricht dabei ungefähr einem Threshold d.h. der Position der Ebene (nicht komplett wahr, aber wahr genug)
- Der Parameter w bestimmt die Ausrichtung der Ebene; er steht senkrecht auf der Ebene (=Normalenvektor) und alle anderen Vektoren im Raum X die orthogonal zu ihm sind spannen gemeinsam die (Hyper-)Ebene auf
- Seine Skalierung ist beliebig, da er nur die Richtung angibt (wir kommen später darauf zurück)
- W und b sind beide Unbekannte, die wir bestimmen wollen
- Nun haben wir die Ebene definiert. Aber wir suchen die optimal partitionierende Ebene nach unserem Optimalitätskriterien (der maximale Abstand/Margin)

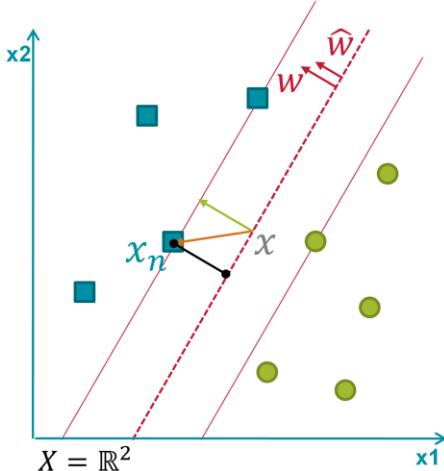
SVM: Definition des Abstandes



25

- Dafür benötigen wir erst einmal eine Definition des Abstandes der Ebene zu den Trainingsobservationen (diesen wollen wir schließlich maximieren)
- Erst einmal zurück in den 2-dimensionalen Observationsraum (alles ist anschaulicher hier, aber gültig für beliebig-dimensionale Räume)
- Nehmen wir eine beliebige Trainingsobservation x_n
- Und die Ebenenformel
- Bestimmen wir den Einheitsvektor \hat{w} (da w ja beliebig skaliert ist)
- Dann den Differenzvektor zwischen x_n und einem beliebigen Punkt x auf der Ebene
- Nur projizieren wir diesen Differenzvektor auf die Richtung des Einheitsvektors, der gleichzeitig der Normalenvektor der Ebene ist
- Der Abstand zwischen der Ebene und der Trainingsobservation x_n ist dann die Länge dieses Vektors

SVM: Definition des Abstandes



Abstand

$$|\hat{w}^T(x_n - x)|$$

Einheitsvektor

$$\hat{w} = \frac{w}{\|w\|}$$

Abstand

$$\frac{1}{\|w\|} |w^T x_n - w^T x|$$

Ebene

$$w^T x + b = 0$$

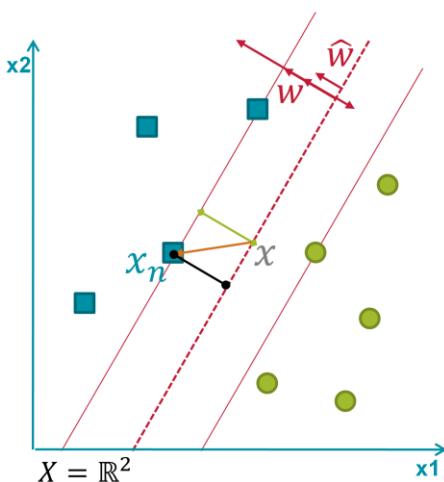
Abstand

$$\frac{1}{\|w\|} |w^T x_n + b - w^T x + b|$$

26

- Das ist keine schöne Formel zur Optimierung, wir versuchen sie zu vereinfachen
- Dafür müssen wir erst einmal in die andere Richtung gehen
- Wir ersetzen den Einheitsvektor \hat{w} durch seine Definition
- Die Ebenenformel beinhaltet den Faktor b . Diesen können wir hinzufügen
- Aber jetzt ist die Formel komplizierter als zuvor!

SVM: Definition des Abstandes



Abstand

$$\frac{1}{\|w\|} |w^T x_n + b - w^T x + b|$$

Ebene

$$w^T x + b = 0$$

Abstand

$$\frac{1}{\|w\|} |w^T x_n + b - 0|$$

Festlegung (für nächstes x_n)

$$w^T x_n + b = 1$$

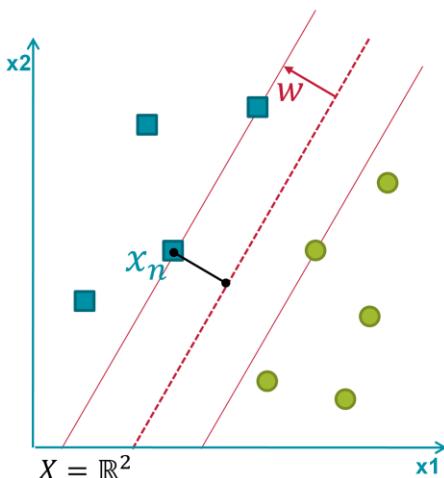
Abstand

$$\frac{1}{\|w\|} |1 - 0| = \frac{1}{\|w\|}$$

27

- Schauen wir uns wieder die Ebenenformel an
- Das entspricht dem rechten inneren Teil unseres Abstandes: substituieren
- Erinnert ihr euch, dass w beliebig skalierbar und nur die Richtung von Relevanz ist?
- Das nutzen wir aus und legen fest, dass die Ebenenformel am Punkt der nächstliegenden Trainingsobservation den Wert 1 liefern soll
- Damit können wir den Abstand weiter vereinfachen und erhalten ein einfacheres Optimierungsproblem

SVM: Das Optimierungsproblem



Abstand / Margin maximieren

$$\max \frac{1}{\|w\|}$$

Randbedingung

$$|w^T x_n + b| \geq 1$$

wobei $n = 1, \dots, N$

28

- Unser Ziel: $1/\|w\|$ zu maximieren
- Mit einer Randbedingung, natürlich
- Das ist nicht trivial, aber machbar
- Duales Problem (min in max)
- Karush-Kuhn-Tucker (KKT) beschreiben, wie eine inequality Bedingung zu einer equality Bedingung umformuliert werden kann
- Damit kann die Methode der Langrange-Multiplizierer angewendet werden (eine Strategie um lokale Minima/Maxima einer Funktion mit equality Bedingungen zu finden)
- Interessierte können sich die Vorlesungen im Anhang dieser Präsentation ansehen oder auf Wikipedia nachschlagen
- Ist sehr interessant, würde allerdings den Rahmen dieser Vorlesung sprengen

SVM: Das Optimierungsproblem

- Karush-Kuhn-Tucker (KKT) Bedingungen: *ineq.* -> *equality*
- Lagrangian Multipliers: *Maximierung mit Randbedingungen*
- Quadratic Programming: *Lösung des Optimierungsproblems*

$$\min_{\alpha} \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N y_n y_m \alpha_n \alpha_m \mathbf{x}_n^T \mathbf{x}_m - \sum_{n=1}^N \alpha_n$$

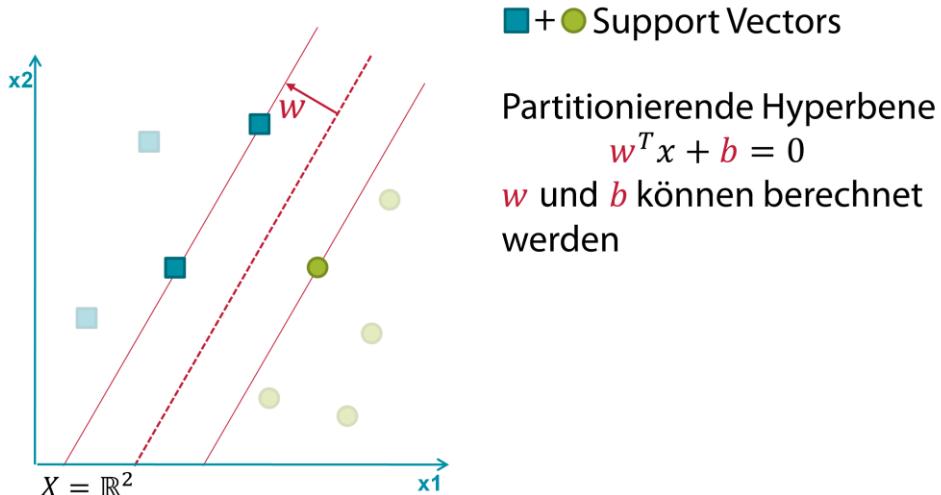
- Komplexität: $\mathcal{O}(N^2)$, bis auf $\mathbf{x}_n^T \mathbf{x}_m$ unabhängig von \mathcal{X}

Ergebnis des Ganzen: Support Vectors / Stützvektoren

29

- Um es kurz zu machen: Wir erhalten dieses Optimierungsproblem
- Die y_s und a_s könnten ignoriert werden, sie sind für diese Vorlesung größtenteils irrelevant
- Wichtig ist, dass die Formel eine quadratische Funktion mehrerer Variablen darstellt
- Diese kann mit QP gelöst werden
- Die Komplexität hängt dabei größtenteils von N ab
- Nur beim Skalarprodukt spielt die Dimensionalität des Observationsraumes eine Rolle
- Als Lösung des Problems erhalten wir einen Vektor a der Länge N
- Und über diesen unsere Support-Vektoren (eine Teilmenge der Trainingsobservationen)

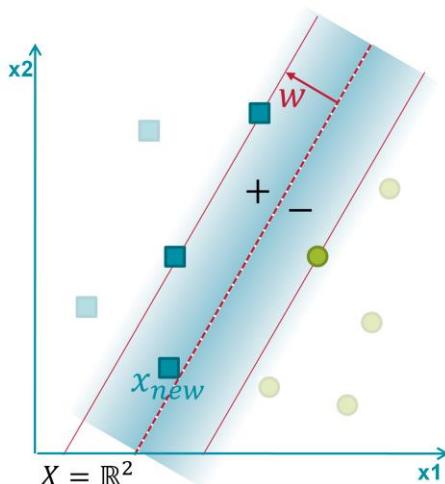
SVM: Support Vectors



30

- Diese Support-Vektoren geben der Methode ihren Namen
- Sie sind die der partitionierenden Hyperebene am nächsten liegenden Trainingsobservationen
- Sie „stützen“ die Ebene
- Da wir wissen, dass die Ebenenformel für sie alle den Wert 1 zurückgeben muss, können wir nun w und b einfach berechnen
- Erhalten die Trainierten SVMs nun eine neue Observation, so muss diese nur in die Ebenenformel eingesetzt werden
- Das Vorzeichen des Ergebnisses bestimmt dann die Klassenzugehörigkeit
- Notiz: SVM sind binäre Klassifikatoren, für mehr als 2 Klassen müssen sie verschachtelt werden
- Notiz: Je weniger Support-Vektoren, desto besser Generalisieren die gelernten SVM d.h. desto kleiner der zu erwartende Fehler

SVM: Der Klassifikator



■ + ● Support Vectors

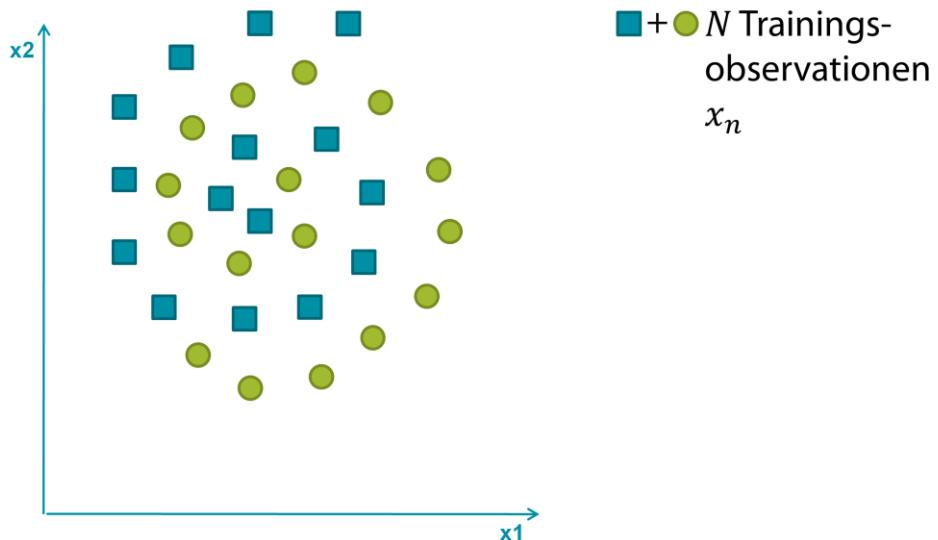
Partitionieren Hyperebene
 $w^T x + b = 0$
 w und b können einfach berechnet werden

Neue Observation x_{new}
 $\text{sgn}(w^T x_{new} + b)$

31

- Diese Support-Vektoren geben der Methode ihren Namen
- Sie sind die der partitionierenden Hyperebene am nächsten liegenden Trainingsobservationen
- Sie „stützen“ die Ebene
- Da wir wissen, dass die Ebenenformel für sie alle den Wert 1 zurückgeben muss, können wir nun w und b einfach berechnen
- Erhalten die Trainierten SVMs nun eine neue Observation, so muss diese nur in die Ebenenformel eingesetzt werden
- Das Vorzeichen des Ergebnisses bestimmt dann die Klassenzugehörigkeit
- Notiz: SVM sind binäre Klassifikatoren, für mehr als 2 Klassen müssen sie verschachtelt werden
- Notiz: Je weniger Support-Vektoren, desto besser Generalisieren die gelernten SVM d.h. desto kleiner der zu erwartende Fehler

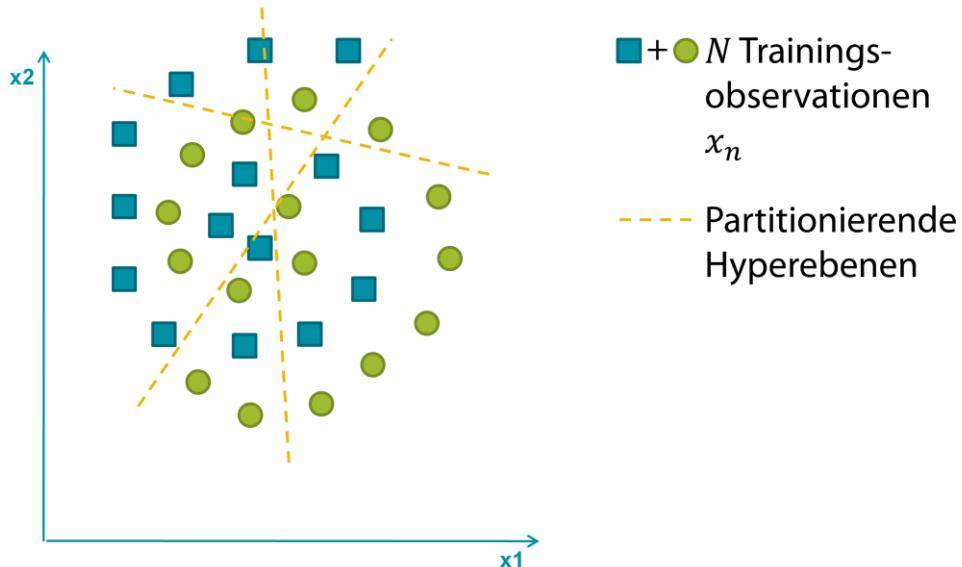
SVM: Nicht-lineare Klassifizierungsprobleme



32

- Klingt einfach, aber was, wenn wir vor einem solchen Problem stehen?

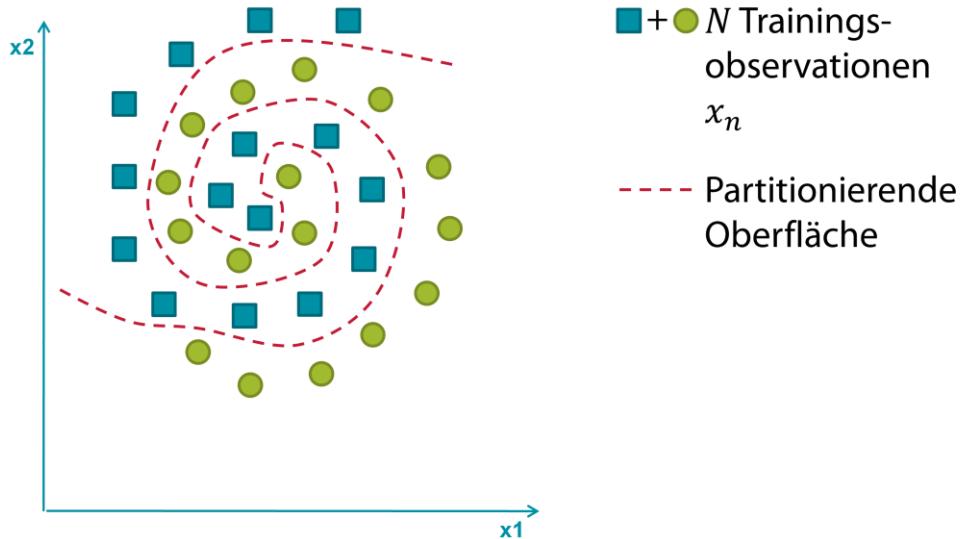
SVM: Nicht-lineare Klassifizierungsprobleme



33

- Welche Linie/Ebene durch die Trainingsmenge generalisiert nun am besten?
- Welche Linie/Ebene partitioniert überhaupt die Trainingsmenge korrekt?
- Genau: Keine!

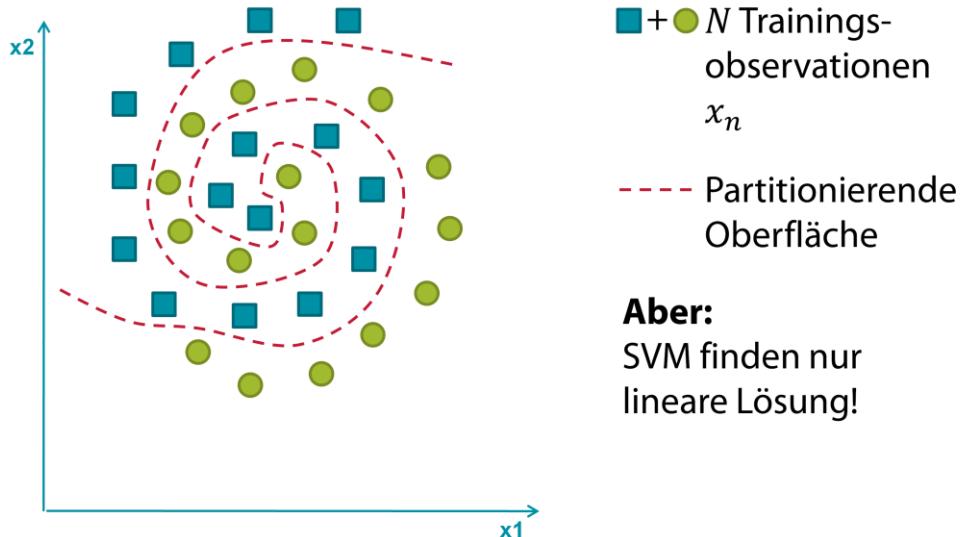
SVM: Nicht-lineare Klassifizierungsprobleme



34

- Was wir bräuchten, wäre etwas die dies: Eine partitionierende Oberfläche.
- Eine solche zu beschreiben wäre allerdings ungleich komplizierter und kaum zu optimieren.

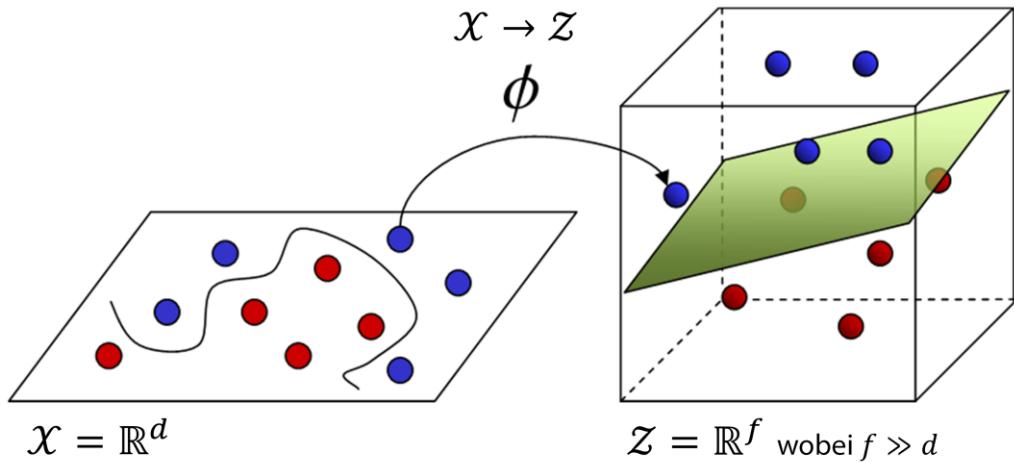
SVM: Nicht-lineare Klassifizierungsprobleme



35

- Unsere SVM Methode kann nur lineare Klassifizierungsprobleme lösen
- Die meisten Probleme sind aber nicht linear
- Also alles umsonst?

SVM: Transformation in den \mathcal{Z} -Raum



Wobei z.B. $\phi(x) = (x_1, x_2, x_1 + x_2)$

Bilder von: www.sbaban.org

36

- Lösung: Transformation in den \mathcal{Z} -Raum
- Ein schöner Trick, nicht?
- Aber was bedeutet das in der Praxis?



SVM: Das Optimierungsproblem

$$\min_{\alpha} \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N y_n y_m \alpha_n \alpha_m \mathbf{x}_n^T \mathbf{x}_m \sum_{n=1}^N \alpha_n$$

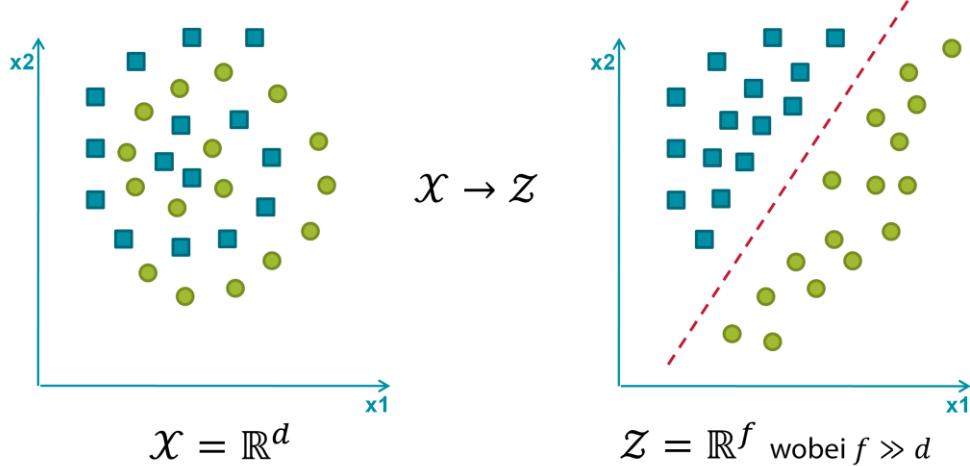
Komplexität: $\mathcal{O}(N^2)$, bis auf $\mathbf{x}_n^T \mathbf{x}_m$ unabhängig von \mathcal{X}

$$\min_{\alpha} \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N y_n y_m \alpha_n \alpha_m \mathbf{z}_n^T \mathbf{z}_m \sum_{n=1}^N \alpha_n$$

Komplexität: Dimensionalität von \mathcal{Z} beeinflusst nur $\mathbf{z}_n^T \mathbf{z}_m$

- Hier unser altes Optimierungsproblem...
- ... und die zugehörige Komplexität.
- Hier das neue Problem...
- ...und die Komplexität steigt nur minimal.
- D.h. die Transformation in einen höherdimensionalen Raum bekommen wir sehr billig.

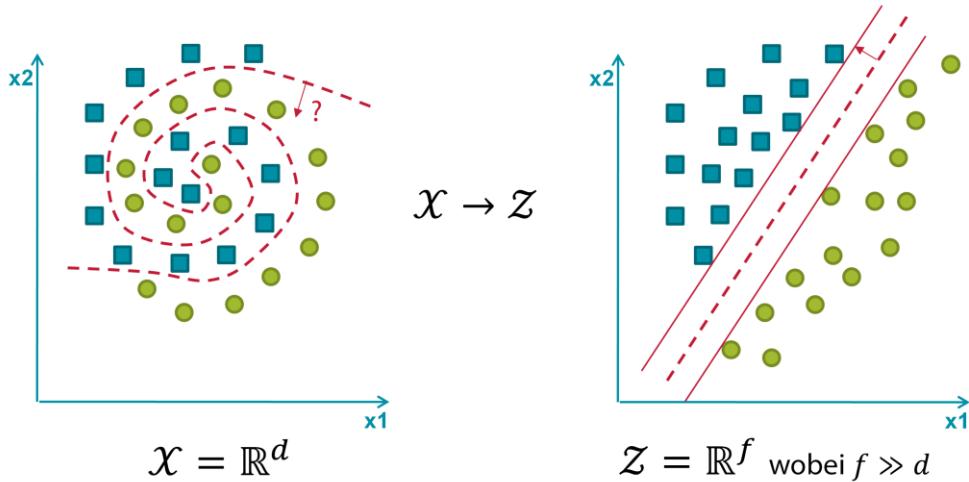
SVM: Transformation in den \mathcal{Z} -Raum



38

- Hier ein schematisches Beispiel, wobei der \mathcal{Z} -Raum nur 2-dimensional dargestellt wird

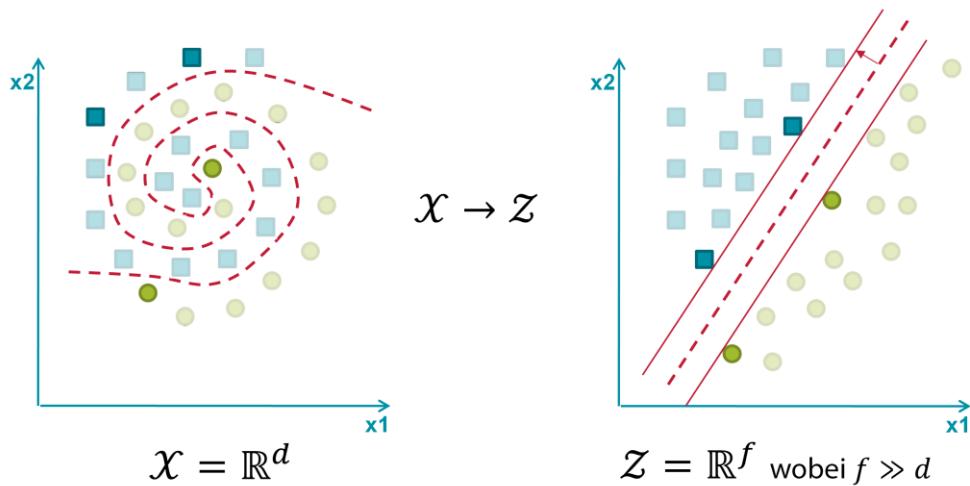
SVM: Transformation in den \mathcal{Z} -Raum



39

- Im ursprünglichen Observationsraum X entspricht der (linear) partitionierenden Hyperebene im Z -Raum eine gebogene Oberfläche
- Wichtig ist: Der Margin wird im Z -Raum festgelegt und bleibt dort, genauso wie die Hyperebene
- Es gibt im ursprünglichen Observationsraum X keine Entsprechung

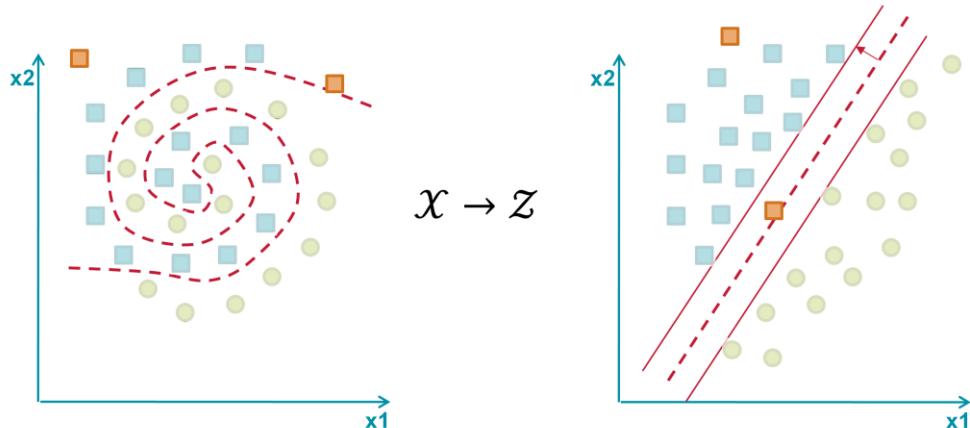
SVM: Transformation in den \mathcal{Z} -Raum



40

- Das geht noch weiter: Die Stützvektoren im \mathcal{Z} -Raum sind die erwartet
- Sie können allerdings völlig beliebig im ursprünglichen Observationsraum \mathcal{X} verteilt liegen und bieten keinerlei Anschaulichkeit

SVM: Klassifizierung im Z -Raum



Skalarprodukt $w^T z$ resp. $z^T z$ muss berechnet werden

41

- D.h. neue Observationen, die klassifiziert werden sollen, werden erst in den Z -Raum transformiert und dann ihre Klasse bestimmt!
- Ihr Abstand zur gekrümmten Oberfläche in X sagt nichts über ihren Abstand zur partitionierenden Hyperebene in Z aus
- Für jede Klassifizierung muss einmal die Ebenenformel in Z gelöst werden, was faktisch der Berechnung eines Skalarproduktes im Z -Raum entspricht



SVM: Vorläufige Zusammenfassung

- Hypothese: Maximaler Margin entspricht bester Generalisierung
- SVM findet partitionierende Hyperebene mit maximalem Margin
- Komplexität des SVM Trainings https. abhängig von N
- SVM kann nur lineare Separierung
- Transformation der Trainingsdaten in höherdimensionalen Raum
- Dort kann eine lineare Separierung gefunden werden
- Erfordert N^2 Skalarprodukte während des Trainings und jeweils eines pro Klassifizierung in \mathcal{Z} -Raum

42

- Wichtig hier: Die Skalarprodukte im Z-Raum



SVM: Die Magie

- Die Berechnung des Skalarproduktes ist vergleichbar schnell
 - **Aber:** lineare Separierbarkeit nur bei sehr hoch-dimensionalem Z -Raum garantiert
 - Komplexität des Skalarproduktes: $\mathcal{O}(\dim(Z))$
 - Training: $z_n^T z_m$
 - Anwendung: $z^T z$
- => Nur das Skalarprodukt, nicht z wird benötigt
- Gibt es eine Möglichkeit das Skalarprodukt implizit zu berechnen ohne die eigentliche Transformation auszuführen?

43

- Alles ok solange $\dim(Z) \ll N$
- Aber was wenn nicht? Was wenn $\dim(Z) \sim N$ oder sogar $\dim(Z) > N$?
- Dann wird die Komplexität des Trainings sowie der Klassifizierung von $\dim(Z)$ bestimmt.
- Können wir dies irgendwie verhindern?
- Schauen wir es genauer an:
 - Wir benötigen das Ergebnis von Skalarprodukten im Z -Raum
 - z, d.h. die Projektion von x nach Z durch ϕ wird nicht benötigt
- Das wirft diese Frage auf

SVM: Der Kernel-Trick

- $\mathcal{X} = \mathbb{R}^2, \mathcal{Z} = \mathbb{R}^6, x = (x_1, x_2), x' = (x'_1, x'_2)$

- $z = \phi(x) = (1, x_1, x_2, x_1^2, x_2^2, x_1 x_2)$

- $z^T z' = \phi(x)\phi(x')$

- $z^T z' = 1 + x_1 x'_1 + x_2 x'_2 + x_1^2 x'^2_1 + x_2^2 x'^2_2 + x_1 x'_1 x_2 x'_2$

- $K(x, x') = (1 + x^T x')^2$

- $K(x, x') = (1 + (x_1 x'_1 + x_2 x'_2))^2$

- $K(x, x') = 1 + [2x_1 x'_1 + 2x_2 x'_2 + x_1^2 x'^2_1 + x_2^2 x'^2_2 + 2x_1 x'_1 x_2 x'_2]$

44

- Ja, gibt es. Die Methode nennt sich der Kernel Trick.
- Ein Beispiel: Gehen wir einmal von folgenden Definitionen aus
- Und bestimmen einen Raum Z durch eine Transformation phi(x) wir folgt
- Projizieren wir also zwei Observationen x und x' in den Z-Raum und bilden ihr Skalarprodukt
- Nun bestimmen wir eine „beliebige“ Funktion K(x,x'), z.B. so wie hier
- Setzen ein, berechnen das Skalarprodukt in X
- Und expandieren ... fällt etwas auf?
- Beide sind äquivalent!
- Bis auf ein die 2en, die uns aber nicht weiter interessieren.
- Diese Funktion K(x,x'), die einer Berechnung des Skalarproduktes in einem anderen Raum Z entspricht , nennt sich Kernel
- Leider ist es nicht trivial von der Formel des Skalarproduktes im Z-Raum auf den entsprechenden Kernel zu schließen
- Hier scheint das möglich: aber stellt euch einen 100-dimensionalen Raum vor



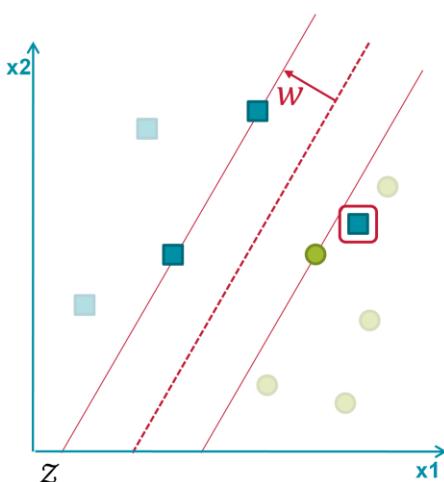
SVM: Kernels

- Müssen mathematische Eigenschaften erfüllen
- Einige bekannte Kernel Funktionen
 - Linear $K(x, x') = xx'$
 - Polynomial $K(x, x') = (\gamma * xx' + r)^d$
 - Radiale Basisfunktion $K(x, x') = \exp(-\gamma|x - x'|^2)$
 - Sigmoid $K(x, x') = \tanh(\gamma * xx' + r)$
- Berechnung auf x , d.h. völlig unabhängig von $\dim(\mathcal{Z})$
- Der \mathcal{Z} -Raum ist unbekannt
- Der \mathcal{Z} -Raum der RBF ist unendlich
- $\exp\left(-\frac{1}{2}|x - x'|^2\right) = \sum_{j=0}^{\infty} \frac{(x^T x')^j}{j!} \exp(-\frac{1}{2}|x|^2)\exp(-\frac{1}{2}|x'|^2)$

45

- Netterweise können ausgedachte Kernels anhand von mathematischen Eigenschaften dahingehend überprüft werden, ob sie den Skalarprodukten in einem höherdimensionalen Raum Z entsprechen
- Das haben (Theoretiker) getan und einige anpassbare Kernelfunktionen gefunden
- Diese sind alle absolut unabhängig von der Dimensionalität des Z -Raumes
- Ja, dieser Raum **ist nicht einmal bekannt**
- Es ist ausreichend, dass die Kernels einem Skalarprodukt in **irgendeinem** Raum Z entsprechen
- Bestes Beispiel für die Magie der Methode: Der RBF Kernel
- Der zugehörige Z -Raum ist von **unendlicher** Dimensionalität
- D.h. es wäre unmöglich das Skalarprodukt dort (in endlicher Zeit) zu berechnen
- Und mit Hilfe des Kernels können wir es trotzdem tun.
- Und somit Klassifizierungsprobleme beliebiger Komplexität lösen.
- Bemerkung: Das bedeutet nicht, dass es sich um eine gute Lösung handeln wird!

SVM: Soft-Margin



- Anzahl der Support-Vektoren bestimmt Fehler
- Ausreißer / Rauschen erhöht benötigte Support-Vektoren
- **Lösung:** Soft-Margin
- Neuer Parameter C
 - Je kleiner C, desto „weicher“ der Margin
- Non-Margin Support-Vektoren

46

- Je mehr Support-Vektoren, desto restriktiver und weniger generalisierend die partitionierende Hyperebene und desto größer der zu erwartende Fehler
- Ausreißer bzw. Rauschen können dazu führen, dass sehr viel mehr Support-Vektoren benötigt werden und der gelernte Klassifikator nur sehr schlecht generalisiert
- Dies können wir verhindern, indem wir die harten Margin-Bedingungen etwas aufweichen
- Wir erlauben einigen wenigen Trainingsobservationen innerhalb des Margin zu liegen
- Wie stark dies erlaubt ist bestimmt ein Weichheitsparameter C
- Diese Trainingsobservationen werden Non-Margin Support-Vektoren genannt und sind Teil der Definition der SVM (neben den klassischen Support-Vektoren)



SVM: Zusammenfassung

- Soft-Margin Klassifikator berücksichtigt Rauschen
- Observationen werden in ∞ -dimensionalen Raum abgebildet
- Nutzt linear Methode um nicht-lineare Klassifizierungsprobleme in einem höherdimensionalen Raum mit dem Kernel-Trick zu lösen
- Komplexität des Trainings abhängig von der Anzahl Trainingsobservationen
- Komplexität der Klassifizierung abhängig von der Anzahl der Support-Vektoren
- Generalisierbarkeit / Zu erwartender Fehler abhängig von der Anzahl der Support-Vektoren
- Black-box Klassifizierer

47

- Black-Box Klassifizierer: Die SVM lernen etwas, aber wir wissen nicht was. Nur anhand ihrer Klassifizierungsergebnisse können wir sie bewerten. Wir können nicht in sie hineinschauen und ihre Entscheidungswege nachvollziehen.
- Gemeinsamkeit mit Neuronalen Netzwerken, zu denen es eine Geschichte gibt:

„Zu Zeiten der großen Maschinellen Lernen Euphorie in den 70-80ern hat die DARPA einige Millionen in die Entwicklung eines Künstlichen Neuronalen Netzwerkes (ANN) gesteckt, das zwischen Fotos mit (versteckten) Panzern darauf und solchen ohne unterscheiden soll. Quasi ein Panzerdetektor.

Das Militär lieferte die Bilder, die Wissenschaftler planten & programmierten und am Ende stand ein ANN, dass die zurückgelegte Evaluationsmenge mit einer hohen Genauigkeit richtig Kategorisieren konnte.

Wissenschaftler sowie Militärs feierten die große Errungenschaft und entwickelten Pläne, wo dieses Netzwerk eingesetzt werden sollte. Doch bald kam die Ernüchterung: Während einiger Praxiseinsätze zeigte das ANN keinerlei Interesse daran Panzer zu erkennen, sondern klassifizierte wild und scheinbar zufällig darauf los. Mehr Tests folgten und schließlich wurde das Projekt begraben. Lange wusste niemand, was eigentlich passiert war.

Erst Jahre später begutachtete ein junger Wissenschaftler die Trainingsbilder erneut und stellte fest, dass die Militärs die Fotos an zwei verschiedenen Tagen aufgenommen hatten: An einem die ohne Panzer und an einem die mit Panzer. Und wie der Zufall wollte schien an einem die Sonne und an anderem nicht...“

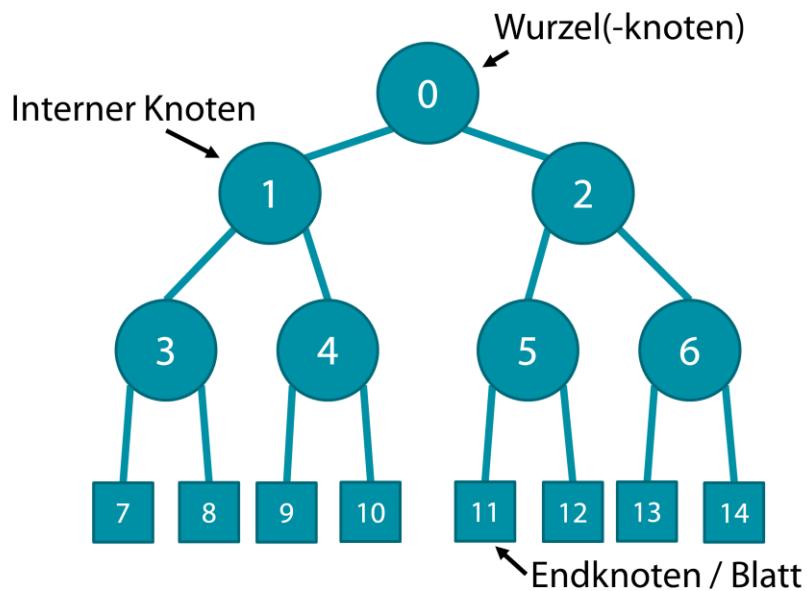


SVM: Q&A



Entscheidungs- bäume

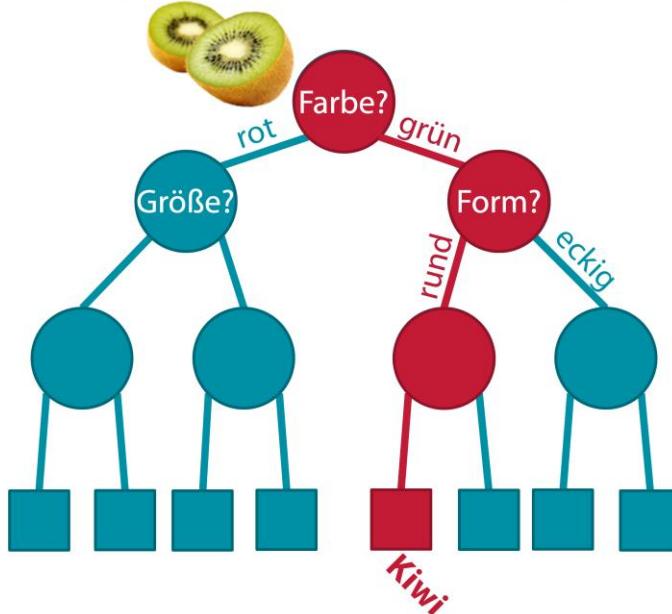
Entscheidungsbaum: Struktur



50

- Beginnt immer mit einem Wurzelknoten, dann kommen beliebige viele Interne Knoten mit jeweils einem Eltern- und zwei Kindsknoten. Die Blätter bilden schließlich die Enden.
- Eigentlich ist es nur ein baum, wenn man ihn sich kopfüber vorstellt. Und auch dann eher ein Busch.

Entscheidungsbaum: Einfaches Beispiel



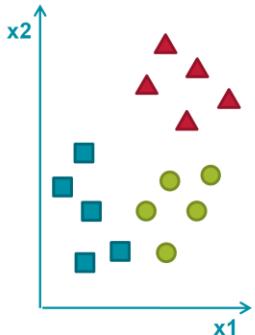
51

- Wir haben eine Frucht und wollen sie Klassifizieren.
- Beginnend an der Wurzel führen wir den binären Test aus und bewegen uns nach rechts.
- Am nächsten Knoten betrachten wir die Form, die eher rundlich ist. Also nach links.
- Noch ein Test und wir erreichen ein Blatt in dem „Kiwi“ steht. Der Baum hat unsere Frucht also (korrekterweise) als Kiwi kategorisiert.

Entscheidungsbaum: Training

- Trainingsobservationen
- $S = \{x_0, \dots, x_N\}$

$$\downarrow S_0 = S$$

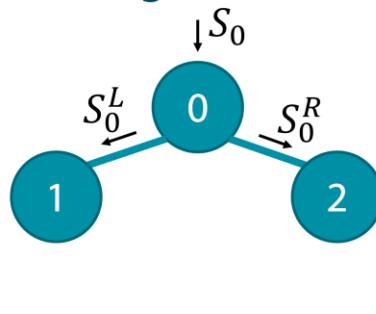
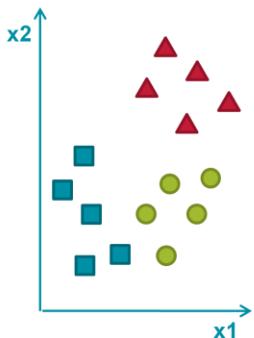



52

- Der vorherige Baum war natürlich ausgedacht. Wir jedoch wollen Bäume automatisch trainieren.
- Wir beginnen wieder mit einer Trainingsmenge S , in diesem Fall mit Beispielen von drei Klassen
- Diese Trainingsmenge geben wir an den Wurzelknoten mit der Nr. 0 und nennen sie S_0 .

Entscheidungsbaum: Training

- Trainingsobservationen
- $S = \{x_0, \dots, x_N\}$

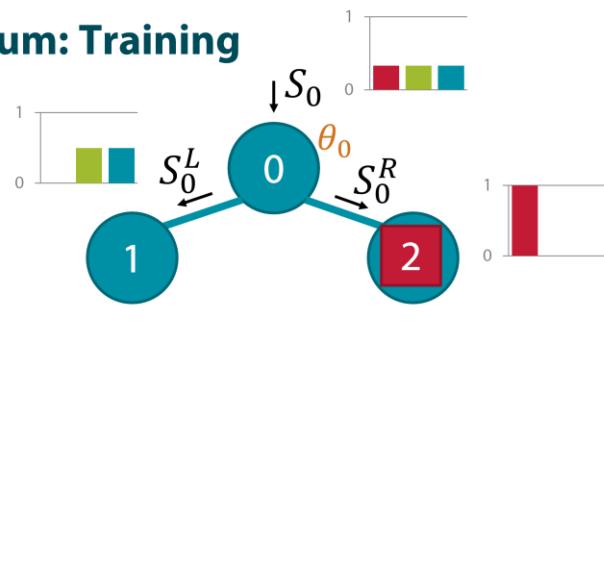
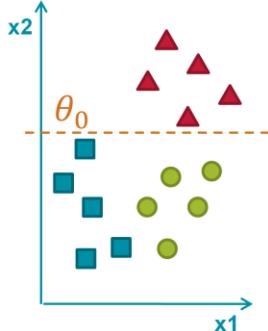


53

- Am Knoten 0 entscheiden wir uns für eine Partition, trennen die Trainingsmenge S_0 in zwei disjunkte Mengen und geben diese an die jeweiligen Kinderknoten weiter.
- Jeder Kinderknoten erhält also nur eine Teilmenge der ursprünglichen Trainingsmenge.

Entscheidungsbaum: Training

- Trainingsobservationen
- $S = \{x_0, \dots, x_N\}$

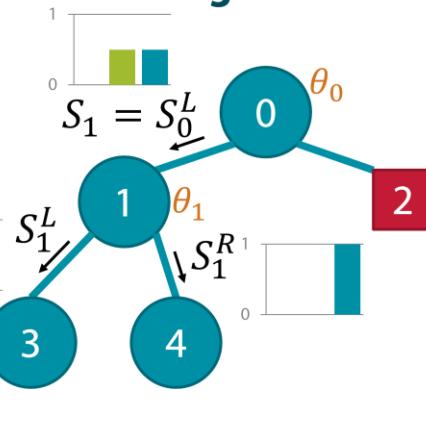
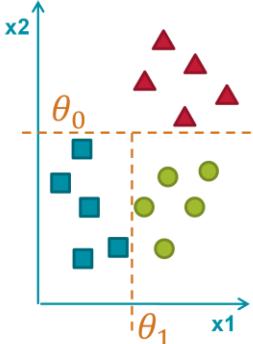


54

- Wenn wir uns die Klassenverteilungen der Mengen vor und nach der Partitionierung ansehen, so können wir beobachten, dass in den Kinderknoten Nr. 2 nur rote Klassenbeispiele gelandet sind.
- Diese Knoten wird darum ein Blatt(-knoten) der Farbe Rot.
- Die Trennlinie durch unsere Trainingsmenge, die der durchgeföhrten Partitionierung entspricht ist diese orangene Linie.
- Wir nennen sie θ . Sie ist definiert durch a) das Merkmal, an dem sie sich ausrichtet (hier x_2) und b) dem threshold (hier wo die Linie auf die x_2 -Achse trifft).

Entscheidungsbaum: Training

- Trainingsobservationen
- $S = \{x_0, \dots, x_N\}$

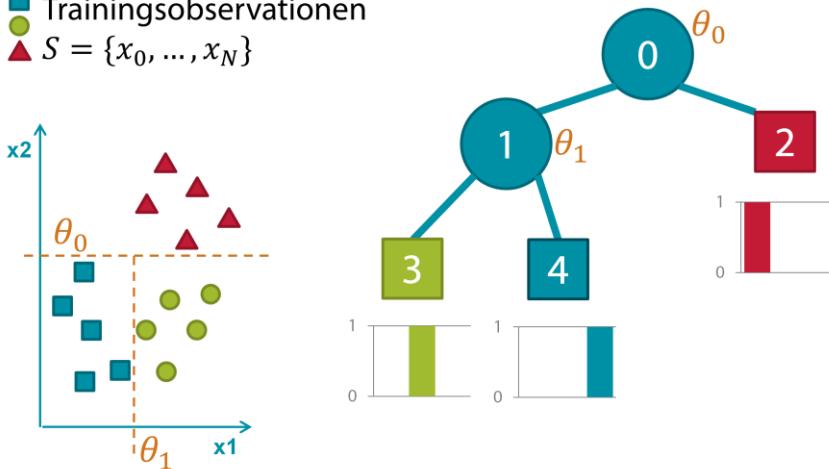


55

- Der Knoten 2 ist ein Blatt und der Baum wächst hier nicht weiter.
- Die Trainingsmenge, die den Knoten 1 erreicht ist noch nicht perfekt partitioniert. Wir wenden also eine weitere Partitionierung an und erhalten die disjunkten Untermengen S_{1_L} und S_{1_R} , die an die Kinderknoten weitergereicht werden.
- Diese Partitionierung entspricht der orangenen Linie \theta_1.

Entscheidungsbaum: Training

- Trainingsobservationen
- $S = \{x_0, \dots, x_N\}$

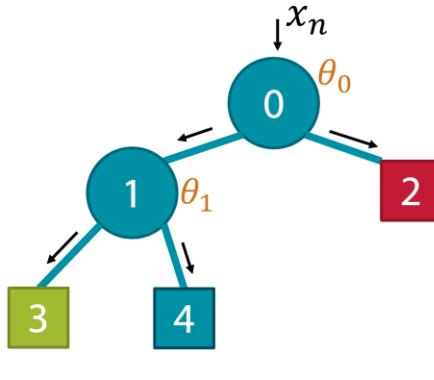
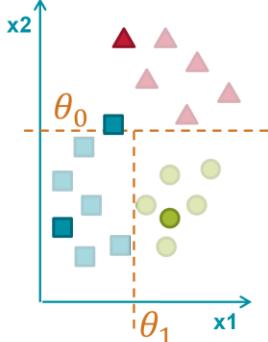


56

- Da sich jeweils nur Observationen einer Klasse in den Set befinden, werden die Knoten 3 +4 zu Blättern mit ihrer jeweiligen Klasse.
- Der Baum ist nun trainiert und Klassifiziert die Trainingsmenge optimal. Er ist bereit für die Anwendung.

Entscheidungsbaum: Anwendung

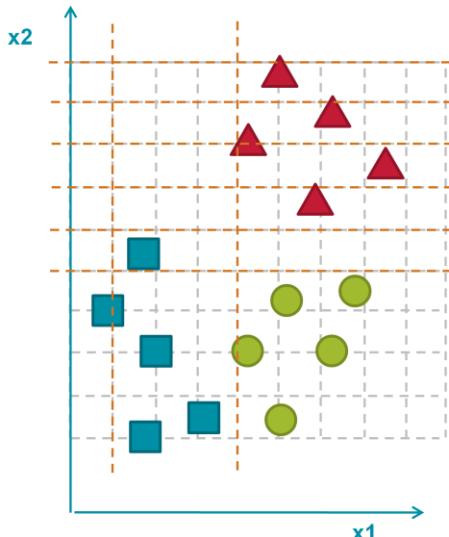
- Trainingsobservationen
- ▲ $S = \{x_0, \dots, x_N\}$



57

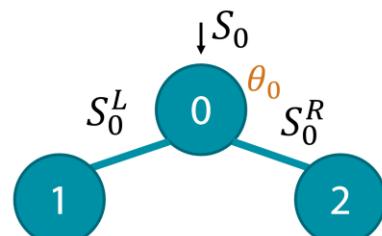
- Hier jeweils ein Beispiel für jede Klasse und den Weg, den die neue Observation durch den Baum nehmen würde.
- Das letzte Beispiel wird fälschlicherweise als der roten Klasse zugehörig klassifiziert. Entscheidungsbäume generalisieren nicht perfekt.

Entropy / Informationsgewinn



Wie finde ich die beste Partition über ein Merkmal?

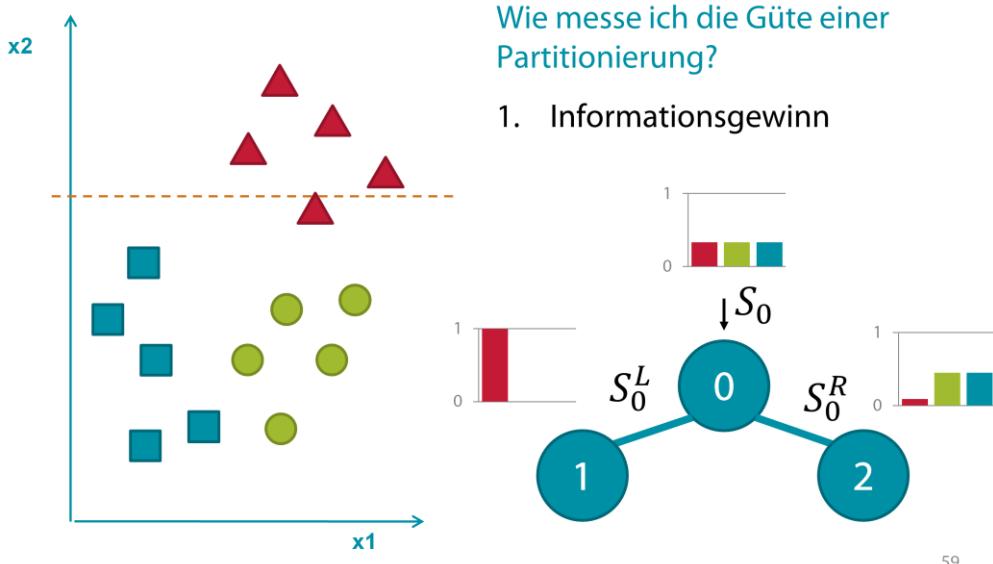
1. Diskretisierung des Merkmalraumes
2. Abtasten



58

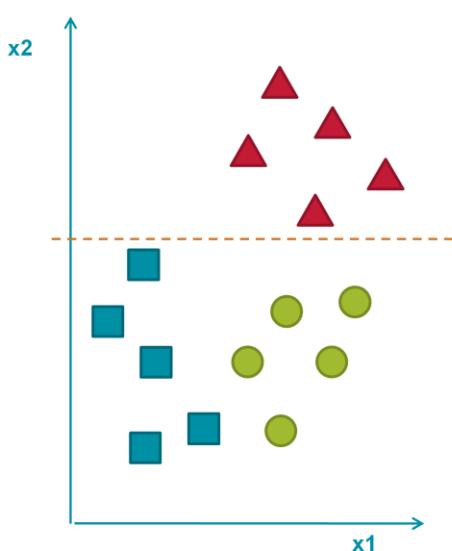
- Ich habe die ganze Zeit von der „besten“ Partitionierung der in jeden Knoten eingehenden Trainingsobservationen geredet. Aber was bedeutet das eigentlich? Wie such ich nach der Besten? Welche stehen überhaupt zur Auswahl?
- An jedem Knoten betrachten wir unseren gesamten Merkmalsraum, wie er links im Diagramm gegeben ist. Und such die achsenparallele Linie, die unsere Menge optimal trennt.
- Da es sich um reelle Merkmale handelt, gibt es unendlich solche möglichen Trennlinien. Um die Aufgabe etwas zu vereinfachen, diskretisieren wir den Suchraum für jedes Merkmal.
- Für das erste Merkmal x_2 durchsuchen wir all diese grauen Linien, so, so und so.
- Für das zweite Merkmal ebenso.
- Und erhalten für jedes Merkmal die optimal partitionierende Linie.
- Nun müssen wir aus diesen nur noch die beste auswählen und wir haben unseren binären Test θ_0 für den Knoten 0.

Entropy / Informationsgewinn



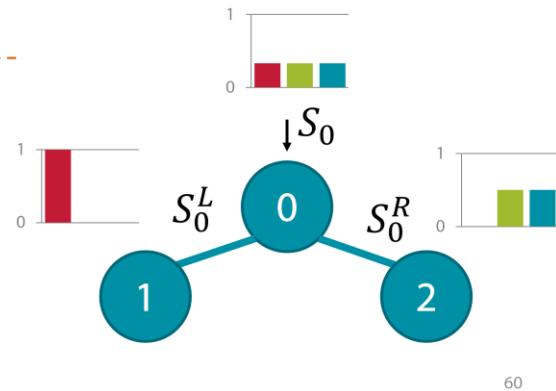
- Wir haben jetzt den Suchraum definiert: Diskretisierte Schwellwerte über alle vorhandenen Merkmale. Aber wie man die Güte einer Partition misst haben wir seither noch nicht geklärt.
- Dafür verwenden wir ein Gütemaß, dass sich Informationsgewinn nennt.
- Hier ein Beispiel: Betrachtet man diese Trennlinie, so existiert an Knoten 1 ein maximaler Informationsgehalt (alle Observationen die hier landen gehören der Klasse Rot an) und an Knoten 2 ein gewisser Informationsgehalt (alle Observation die hier landen gehören wahrscheinlich der Klasse blau oder grün an). In jedem Fall erhöht sich die Information gegenüber dem Knoten 0.

Entropy / Informationsgewinn



Wie messe ich die Güte einer Partitionierung?

1. Informationsgewinn



60

- Nehmen wir stattdessen diese Trennlinie, so änderst sich für die Knoten 0 und 1 nichts. Für Observationen, die den Knoten 2 erreichen, können wir nun aber sicher sein, dass sie entweder zur Klasse Grün oder Blau gehören. Wir haben mehr Information gewonnen als im vorhergehenden Beispiel.



Entropy / Informationsgewinn

Informationsgewinn I

$$I = H(S) - \frac{|S^L|}{|S|}H(S^L) - \frac{|S^R|}{|S|}H(S^R)$$

Shannon-Entropie $H(S)$

$$H(S) = - \sum_{c \in C} p(c) \log_2 p(c)$$

Wobei $C = \{\blacksquare, \bullet, \blacktriangle\}$ und $p(c)$ die Wahrscheinlichkeit c in der Menge S anzutreffen.

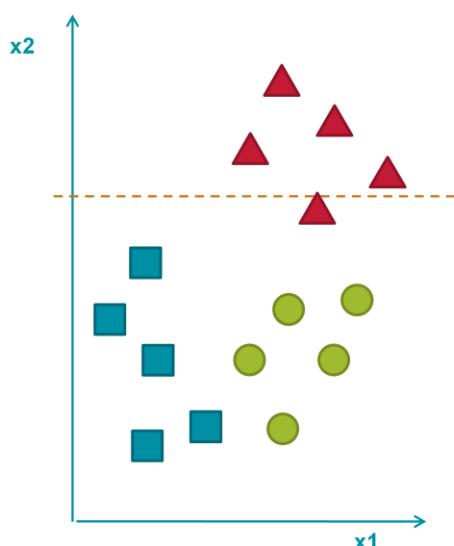
Entropie:

- Maß für den mittleren Informationsgehalt.
- Je geringer die Entropie, desto höher der Informationsgehalt.
- Entropie (\sim Unordnung) nimmt monoton zu.

61

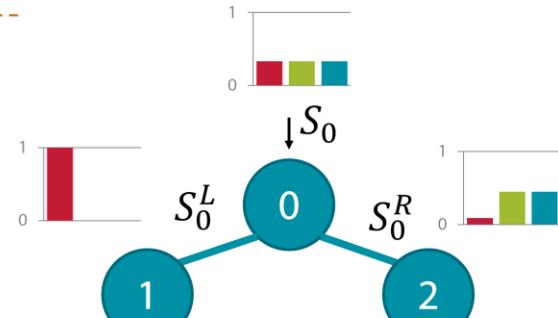
- Wir arbeiten mit Computern, der Computer braucht konkrete Zahlen, mit denen er arbeiten kann.
- Wir definieren den Informationsgehalt als der Unterschied in der Entropie des Trainingssets vor der Partitionierung $H(S)$ und der gewichteten Entropie der Trainingsmengen nach dem Split.
- Die Entropie, die wir hier verwenden ist die Shannon-Entropie der Informationstheorie. Sie ist definiert als die Summe der Multiplikationen der Klassenwahrscheinlichkeiten mit den zugehörigen log-Wahrscheinlichkeiten.
- Entropie ist ein sehr interessantes Maß, dass in vielen Bereichen wie z.B. der Philosophie Verwendung findet. Es misst grob gesagt den Informationsgehalt, ein Konzept, dass genauso schwer zu fassen ist, wie die Idee von „Information“
- In unserem Fall können wir die Entropie ungefähr mit der Unordnung gleichsetzen.
- Per Definition nimmt die Entropie monoton ab. Dass wirft interessante Fragen über unser Universum auf, da es zwangsweise immer geordneter wird.
- Aber zurück zum Thema...

Entropy / Informationsgewinn



Wie messe ich die Güte einer Partitionierung?

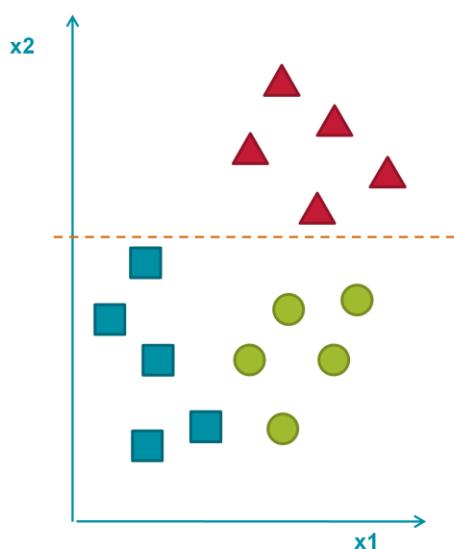
1. Informationsgewinn $I = 0.18$



62

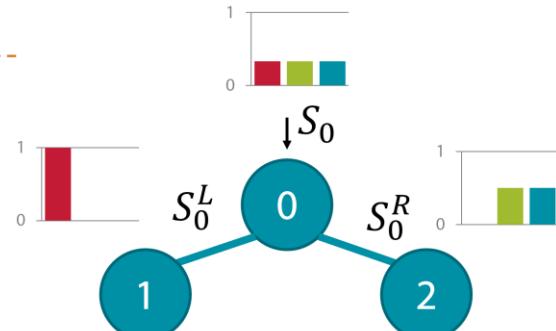
- Berechnen wir den Informationsgewinn für unsere Beispiele sehen wir sofort, welche Partitionierung die bessere ist: Und zwar die, die wir auch intuitiv gewählt hätten.

Entropy / Informationsgewinn



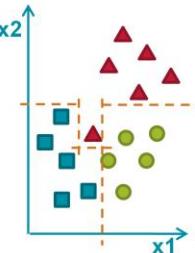
Wie messe ich die Güte einer Partitionierung?

1. Informationsgewinn $I = 0.28$



Entscheidungsbaum: Zusammenfassung

- Automatische, deterministische Klassifizierer
- Optimale Partitionierung nach Trainingsdaten
- „Wichtige“ Merkmale befinden sich höher im Baum
- Sehr abhängig von der Güte der Trainingsdaten
- Hang zur Überanpassung



64

- Entscheidungsbäume sind nur für wenige Klassifizierungsaufgaben geeignet
- Sie haben aber eine menge interessanter Eigenschaften, wie die geringe Größe, die Geschwindigkeit des Trainings und der Anwendung



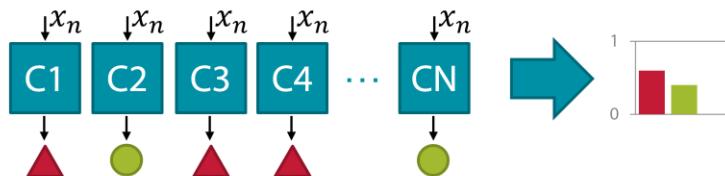
Entscheidungs- Bäume: Q&A



Random Decision Forests (RDF)

Mehr ist besser: Ensemble Methoden

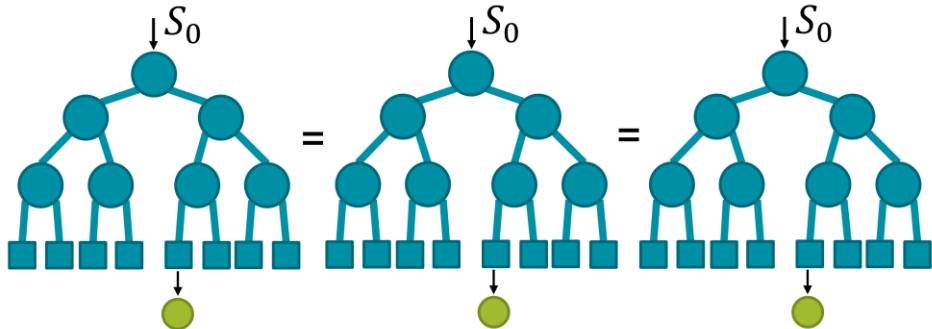
- Bekannte Probleme vieler Klassifizierer
 - Große Abhängigkeit von den Trainingsdaten
 - Überanpassung / Overfitting
 - Empfindlich gegenüber Rauschen und Ausreißern
 - Fluch der Dimensionalität
- Idee: **Ensemble Methoden**
 - Ein ganzes Ensemble sog. schwacher Klassifizierer wird trainiert
 - Abstimmung über die Klassenzugehörigkeit bzw. Wahrscheinlichkeit



67

- Wir haben gerade über die Nachteile der Entscheidungsbäume geredet. Einige Wissenschaftler kamen darüber auf die Idee, immer gemäß dem Leitsatz „Mehr ist besser“, Ensemble Methoden vorzuschlagen.
- Die Idee ist simpel: Wir trainieren eine große Menge kleiner, schneller und nicht so guter Klassifizierer.
- Während der Anwendung geben wir dann die neue Observation an alle von diesen und lassen sie über die richtige Klasse abstimmen.

Random Decision Forests



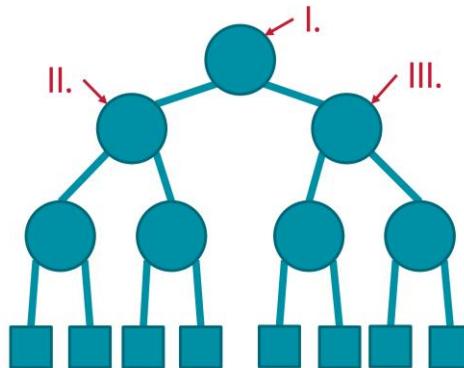
- **Problem:** Entscheidungsbäume trainieren deterministisch
- **Deshalb:** **Random** Decision Forests

68

- Zu dieser Klasse der Methoden gehören auch die Random Decision Forest – und deshalb heißen sie Wälder.
- Wir trainieren viele Entscheidungsbäume und wenden diese dann in parallel an.
- **Aber:** Wenn wir alle Bäume mit den selben Trainingsdaten trainieren, dann sehen sie auch genau gleich aus. Das nimmt ein wenig den Sinn aus der Abstimmung, wenn alle immer für die gleiche Klasse stimmen.
- Hier kommt das Random im Random Decision Forest ins Spiel.

Das Random in Random Decision Forests

- I. Zufällige Auswahl der Trainingsmenge
- II. Zufällige Auswahl einer Untermenge der Merkmale
- III. Zufällige Auswahl des Schwellwertes



69

- Ich werde euch drei Methoden vorstellen um Zufälligkeit zum Training der Forest hinzuzufügen.
- Es gibt noch andere, aber diese sind die populärsten und wahrscheinlich Effektivsten.
- Sie können übrigens alles beliebig kombiniert werden.



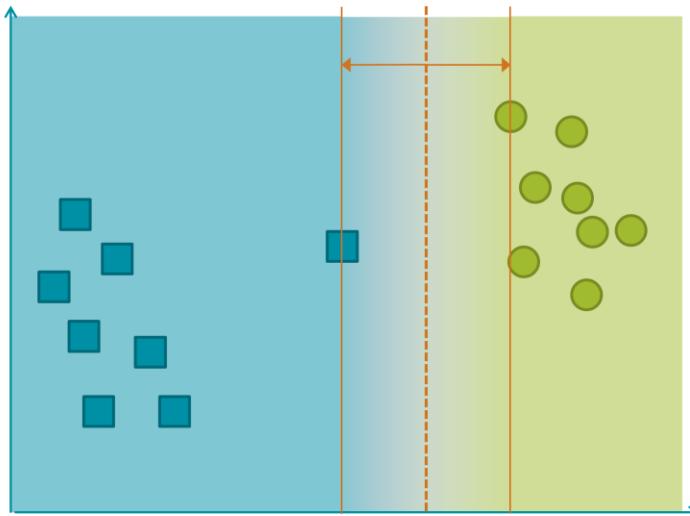
Randomness I: Bagging

- Jeder Baum t wird mit einer zufällig gesampelten Untermenge S_t der Gesamttrainingsmenge S trainiert
- Häufig gleichmäßiges Sampeln mit Zurücklegen
- Jeder Baum wird ein wenig anders
- Große Cluster im Merkmalsraum wirken sich weiterhin stark aus
- Einzeln stehende Observationen werden runtergewichtet

70

- Die erste Randomness ist die einfachste: Bagging
- Es bedeutet einfach, dass wir jedem Baum nicht die gesamte Trainingsmenge zum Training zur Verfügung stellen, sondern nur eine zufällige Untermenge.
- Dadurch trainiert jeder Entscheidungsbaum auf einer leicht unterschiedlichen Menge und sieht damit auch leicht unterschiedlich aus.

Randomness I: Bagging



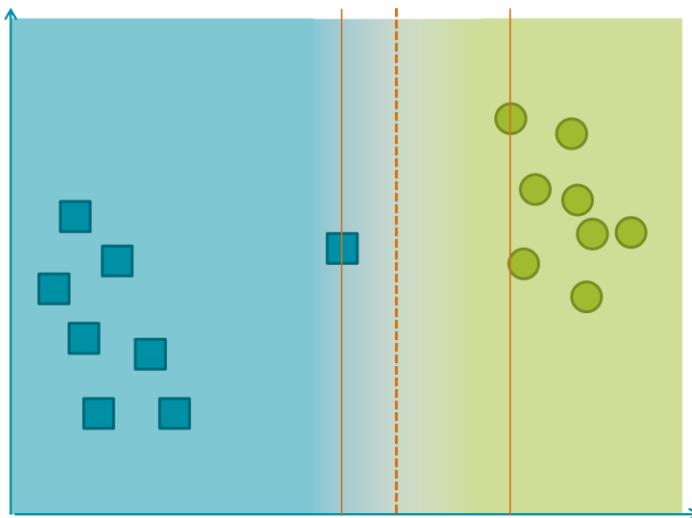
Beispiel ohne
Bagging:

- Wie Maximaler Margin
- Blauer Ausreißer hat großen Einfluss

71

- Die wäre ein Beispiel eines Trainings ohne Bagging (aber mit anderer Zufälligkeit): Alle Bäume definieren ihre Trennlinie in dem Bereich zwischen den beiden durchgezogenen Linien.
- Wenn wir genügend Bäume trainieren, verteilen diese sich gleichmäßig über diesen Bereich. Die Abstimmungsgrenze aller Bäume liegt dann auf der gestrichelten Linie.
- Genau in der Mitte zwischen den nächsten Observationen: Maximum-Margin Klassifizierer, siehe SVMs
- Wie wir schon wissen, ist diese Lösung aber nicht ideal. Der einzelne blaue Ausreißer hat die selbe Gewichtung wie das große Cluster von grünen Observationen.

Randomness I: Bagging



Beispiel mit Bagging:

- Wie Soft-Margin SVM
- Manche Bäume trainieren ohne den Ausreißer
- Ausreißer hat weniger Einfluss

72

- Durch die Anwendung des Bagging werden im Schnitt 1/8 der Bäume eine Trainingsmenge erhalten, in der der Ausreißer nicht vorkommt. Diese Bäume werden ihre Trennlinie sehr viel weiter links wählen.
- Die mittlere Trennlinie über alle Bäume rutscht dadurch ein Stück nach links, so wie wir es gerne hätten.
- Das hat den selben Effekt wie die Soft-Margin SVMs.

Einschub: Formalisierung des Trainings

Binäre Partitionierung

$$h(x, \theta_j) = [\tau > \phi(x)]$$

, wobei ϕ ein einzelnes Merkmal aus einer Observation x extrahiert, τ einen Schwellwert darstellt und $\theta_j = \{\tau, \phi\}$ ihre Menge darstellt.

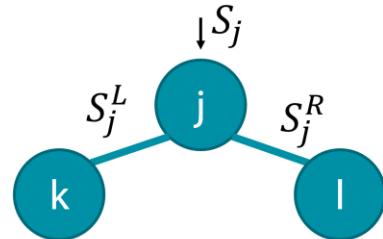
Informationsgewinn

$$I_j = I(S_j, S_j^L, S_j^R, \theta_j)$$

Optimierungsproblem

$$\theta_j^* = \arg \max_{\theta_j \in \Upsilon} I_j$$

, wobei Υ die Menge aller möglichen Parameterkombinationen darstellt.



73

- Wir benötigen noch einmal ein wenig Mathematik bzw. eine Formalisierung des Trainings damit die nächsten Schritte verständlich werden.
- Während des Trainings versuchen wir, wie schon erwähnt, die beste Partition des eingehenden Trainingssets zu finden. Dies stellt ein Optimierungsproblem dar.
- Die Binäre Partitionierung ist definiert als $h(x, \theta_j)$, wobei ϕ einfach den Wert eines Merkmals aus der Observation x extrahiert und dieser daraufhin mit einem Schwellwert τ verglichen wird. Je nach Ausgang dieses Vergleiches wird die Observation den linken oder den rechten Ast weitergeschickt.
- Der Informationsgewinn ist unser Gütemaß, wir definieren ihn hier nur noch einmal ein wenig abstrakter.
- Unser Optimierungsproblem ist dann diese Suche nach dem besten Parameterset θ für den Knoten j über die Menge aller möglichen Parameterkombinationen (aus Merkmal und (diskretisiertem) Schwellwert).

Randomness II: Zufällige Merkmale

Optimierungsproblem

$$\theta_j^* = \arg \max_{\theta_j \in Y_j} I_j$$

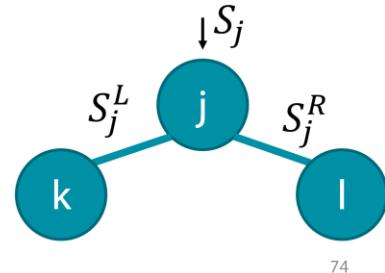
, mit Menge der unbekannten Parameter $\theta_j = \{\tau, \phi\}$.

Zufällige Merkmale

$M = |x|$ ist die Menge aller Merkmale (d.h. Dimensionalität des Merkmalsraumes \mathcal{X}).

$M_j \subset M$ ist eine Untergruppe von ρ zufällig gesampelten Merkmalen für Knoten j .

$Y_j \subset Y$ ist dann die Menge aller möglichen Parameterkombinationen θ_j deren Merkmalselektor ϕ ein $\mu \in M_j$ selektiert.

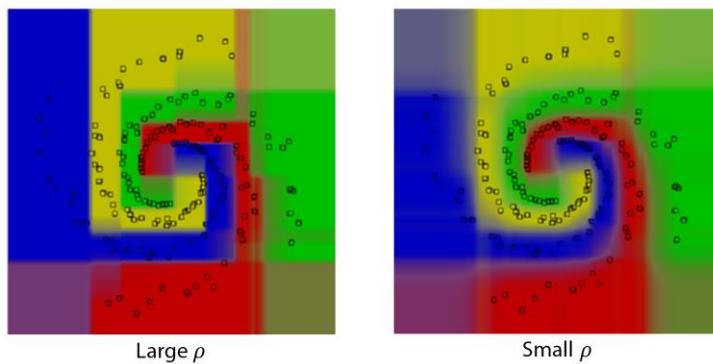


74

- So, zurück zu unser Zufälligkeit...
- Um Zufällige Merkmale zu Verwenden ändern wir unser Optimierungsproblem ein wenig
- Seht ihr den Unterschied? Es ist nur der kleine Index am \Upsilonpsilon.
- Was wir tun und was hier formal und daher ein wenig komplizierter beschreiben ist, ist dass wir anstatt aller Merkmale und ihrer zugehörigen Schwellenwerte nur eine kleinere Untergruppe M_j und zufälligen Schwellwerten berücksichtigen.

Randomness II: Zufällige Merkmale

- Schnelleres Training, da der Suchraum kleiner ist
- Reflektiert verschiedene Herangehensweisen an das Problem
- Robuster gegen schlecht gewählte Merkmale
- Klassengrenzen werden unschärfer => geringere Überanpassung



75

Bilder aus: A. Criminisi et al., "Decision Forests for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning", 28 October 2011, Microsoft Technical Report

- PS: Großes ρ = weniger Zufälligkeit

Randomness III: Zufälliger Schwellwert

Optimierungsproblem

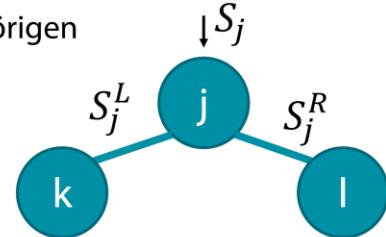
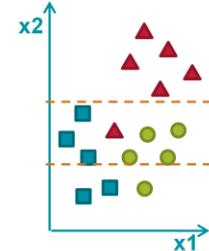
$$\theta_j^* = \arg \max_{\theta_j \in \Gamma} I_j$$

, mit Menge der unbekannten Parameter $\theta_j = \{\tau_\phi, \phi\}$.

Zufälliger Schwellwert

Γ ist die diskrete Menge aller Schwellwerte τ .

Aus den zu jedem Merkmalselektor ϕ gehörigen Schwellwerten $\Gamma_\phi \in \Gamma$ wird ein zufälliger Schwellwert τ_ϕ gewählt.



76

- Die dritte Zufälligkeit, die ihr kennenlernen werden, die der Zufällige Schwellwert
- Dies ist eine sehr starke Zufälligkeit, mit der wir unser Optimierungsproblem stark vereinfachen, indem wir den Suchraum stark einschränken
- Die Idee ist, dass an jedem Knoten nicht eine Anzahl diskretisierter Schwellwerter für jedes Merkmal betrachtet wird, sondern nur ein einziger, zufällig ausgewählter



Randomness III: Zufälliger Schwellwert

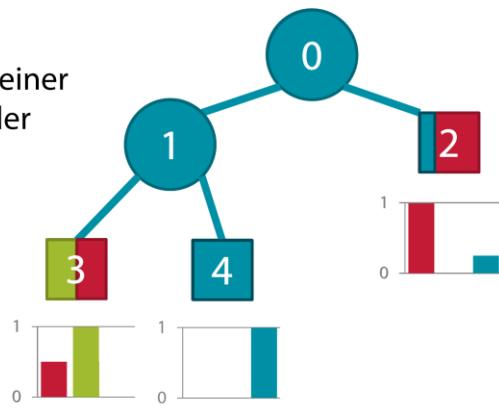
- Schnelleres Training, da sehr viel kleinerer Suchraum
- Sub-optimale und schlechte Partitionen häufig
- Ermöglicht nicht-intuitive Herangehensweisen an das Problem
- Größere / tiefere Bäume
- Robuster gegen Rauschen und Ausreißer

77

- Die Idee hört sich ein wenig gewagt an, funktioniert in der Praxis aber relativ gut, vor allem bei komplizierten Problemen.

Wahrscheinlichkeitsblätter

- Mehrere Bäume stimmen ab => Ausgabe muss nicht mehr „crisp“ sein
- Observationen verschiedener Klassen in einem Blatt erlauben
- Baum gibt Wahrscheinlichkeit einer Klassenzugehörigkeit anstatt der Klassenzugehörigkeit zurück



78

- Die RDFs bringen eine weitere Änderung mit sich: Wo die Entscheidungsbäume zuvor immer weiter gewachsen sind, bis sich nur noch Trainingsobservationen einer Klassenzugehörigkeit in einem Blatt befinden, haben wir jetzt die Möglichkeit das Baumwachstum schon vorher zu beschränken, da die Bäume sowieso gemeinsam abstimmen.
- Wir können z.B. festlegen, dass Mengen von ≤ 5 Observationen nicht weiter gesplittet werden sollen. Oder dass der Baum nicht tiefer als 10 Ebenen wachsen soll.
- Dadurch erhalten wir Wahrscheinlichkeitsblätter, wie hier sichtbar.
- Diese können helfen eine Überanpassung zu verhindern.



Forest-Parameter

size (Größe)

Bestimmt die Anzahl der Bäume im Ensemble (z.B. $T = 200$)

depth (Tiefe)

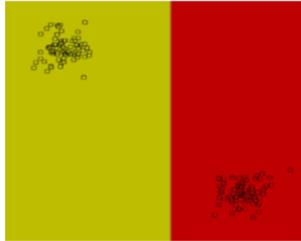
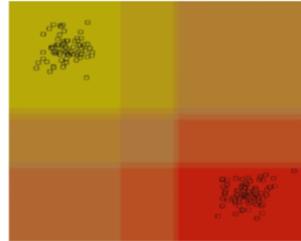
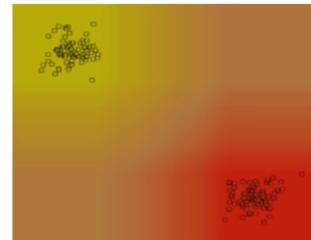
Bestimmt die maximale Tiefe der Bäume. (z.B. $D = 25$)

randomness (Zufälligkeit)

- bagging-size (z.B. $b = 50\%$)
- feature-per-node (z.B. $\rho = \sqrt{|M|}$)

- Der RDF Algorithmus kommt mit einer Anzahl von Paramtern
- Diese sind recht einfach zu setzen und die RDFs sind robust gegen leichte Abweichungen vom Idealwert
- Hier sind einige Parameter mit typischen Werten

Forest-Parameter: size T

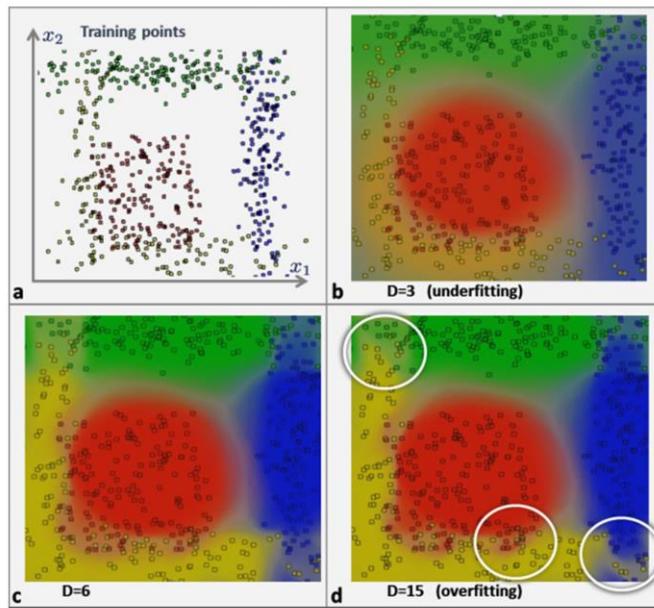
 $T = 1$  $T = 8$  $T = 200$

Bilder aus: A. Criminisi et al., „Decision Forests for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning“, 28 October 2011, Microsoft Technical Report

80

- Die Größe des Waldes wirkt sich wie hier gezeigt aus. Die Bäume in diesem Beispiel sind sog. Stümpfe, d.h. sie haben nur zwei Ebenen, führen also auch nur eine einzige binäre Entscheidung durch.
- Es ist direkt deutlich, dass mehr Bäume einen Klassifizierungsraum liefern, der die Verteilung der Trainingsdaten besser wiederspiegelt und gut Generalisiert.
- Für T einen größeren Wert zu wählen schadet der Klassifizierungsgenauigkeit praktisch nie, macht die RDFs aber größer und verlängert das Training sowie die Anwendung.

Forest-Parameter: depth D

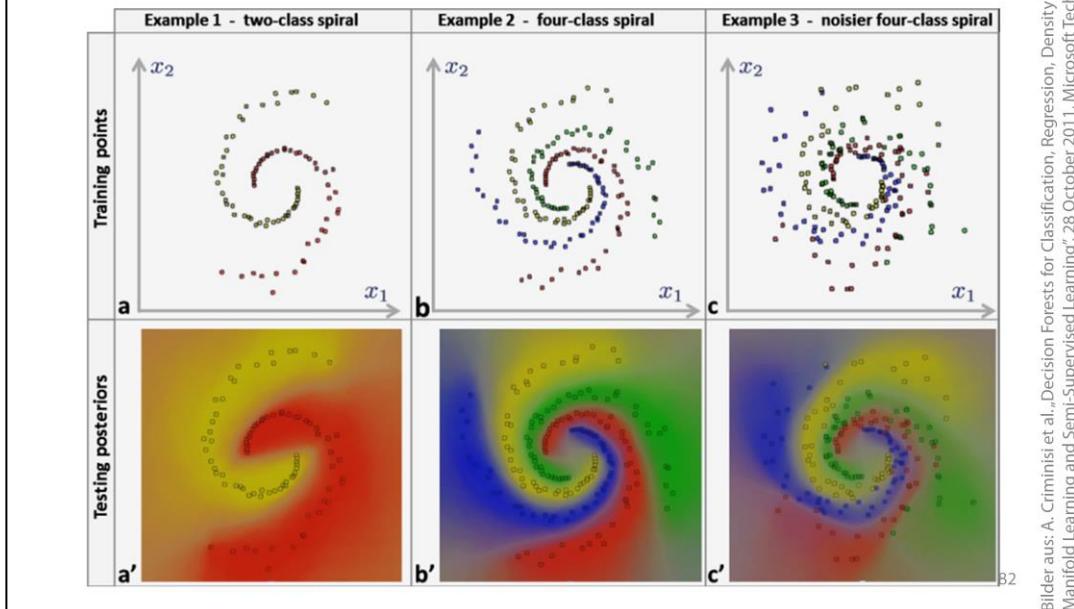


81

Bilder aus: A. Criminisi et al., "Decision Forests for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning", 28 October 2011, Microsoft Technical Report

- Eine Beschränkung der Tiefe der Bäume kann eine Überanpassung verhindern, wie hier dargestellt ist.

RDFs und Rauschen



Bilder aus: A. Criminisi et al., "Decision Forests for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning", 28 October 2011, Microsoft Technical Report

- RDFs sind sehr dankbare Klassifizierer, die auch mit den kompliziertesten Problemen umgehen können.
- Hier das Beispiel einer Spiral (auch Swiss-Roll genannt) aus Observation von vier Klassen. Der RDF Algorithmus lernt die Spirale gut und Generalisiert gleichzeitig. Die partitionierende Oberfläche liegt dort, wo wir sie haben wollen.
- Die rechte Spalte zeigt wie gut RDFs mit Ausreißern / Rauschen umgehen können. Die Trainingsdaten sind eine verrauschte Version des zweiten Beispiels. Die Spiralform ist mit dem bloßen Auge nicht mehr zu erkennen.
- Der RDF schafft es aber den unterliegenden Klassenraum korrekt zu erkennen.



Random Decision Forests: Zusammenfassung

- Ensemble von schwachen Klassifizierern
- Implizit Mehrklassen-Klassifizierer
- Durch Zufälligkeit robust gegen schlechte Merkmale
- Durch Zufälligkeit robust gegen Ausreißer und Rauschen
- Durch Zufälligkeit gute Generalisierung
- Nicht-lineare Klassifizierung
- In Praxis robust gegen Parameterwahl
- Komplexität des Trainings abhängig von der Anzahl der Bäume und der Merkmale
- Komplexität der Klassifizierung abhängig von der Anzahl der Bäume
- Klassifizierung einfach nachzuvollziehen (keine Black-Box)

83

- Ein Unterschied zu SVM ist, dass RDF implizit mit mehreren Klassen umgehen können, während die Verwendung von SVM für mehrere Klassen ein Training von mehreren SVM erfordert.
- Der größte Vorteil der RDF, für mich wenigstens, ist der Folgende: RDFs sind Kritall-Box Klassifizierer. Es ist möglich sich die einzelnen Bäume anzuschauen, jeden Knoten zu betrachten und jeden binäre Split nachzuvollziehen. Ebenso kann während der Anwendung der Pfad der neuen Observation durch die Bäume verfolgt und nachvollzogen werden.
- Fazit: Man fühlt sich einfach wohler mit RFDs, da man das Gefühl bekommt man wüsste, was man vor sich hat.

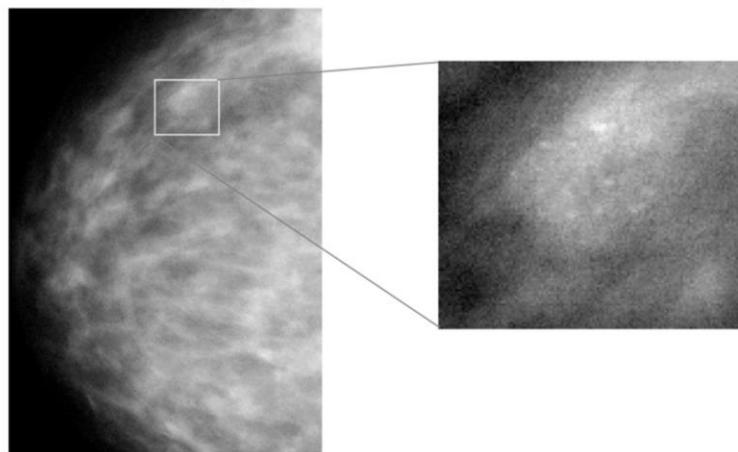


RDF: Q&A

Anwendungsbeispiele: SVM

El-Naqa et al. „A Support Vector Machine Approach for Detection of Microcalcifications“

IEEE Transactions on Medical Imaging 2002



Mammografie & Detailansicht eines MC-Clusters

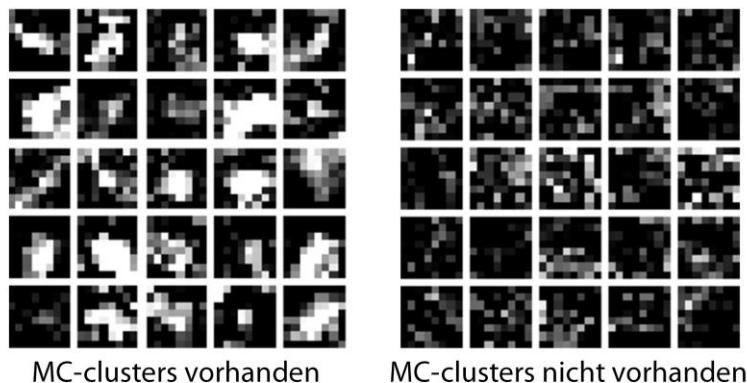
85

- Mikrokalkeinlagerungen treten in Brüsten normal auf und sind selbst unschädlich
- Brustkrebs im Frühstadium kann teilweise daran erkannt werden, dass Mikrokalkeinlagerungen in einer länglichen Anordnung (entlang der Drüsenkanäle) auftreten
- Es ist also sinnvoll für den Arzt, die Mikrokalkeinlagerungen zu identifizieren und ihre Lage zu betrachten
- El-Naqa et al. Haben SVM genutzt, um die schlecht sichtbaren Mikrokalkeinlagerungen automatisch zu finden, damit der Arzt keine Übersicht

Anwendungsbeispiele: SVM

El-Naqa et al. „A Support Vector Machine Approach for Detection of Microcalcifications“

IEEE Transactions on Medical Imaging 2002



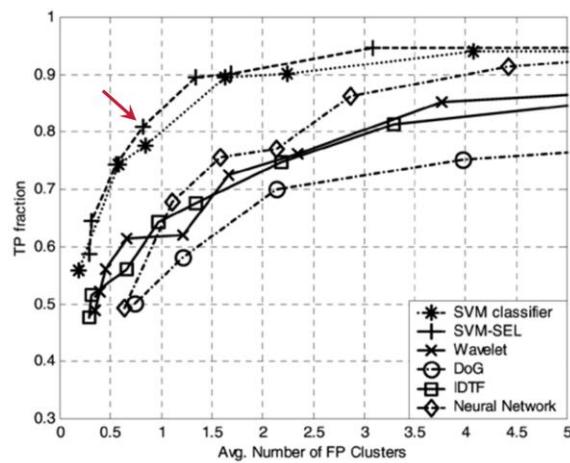
86

- Dies sind Beispiele der verwendeten Trainingsdaten
- Es handelt sich um kleine Bereiche um den zentralen Pixel (der, der Klassifiziert werden soll) herum
- Da die Bereiche 9x9 groß sind, besteht der Merkmalsvektor jeder Observation aus 9x9 Merkmalen

Anwendungsbeispiele: SVM

El-Naqa et al. „A Support Vector Machine Approach for Detection of Microcalcifications“

IEEE Transactions on Medical Imaging 2002



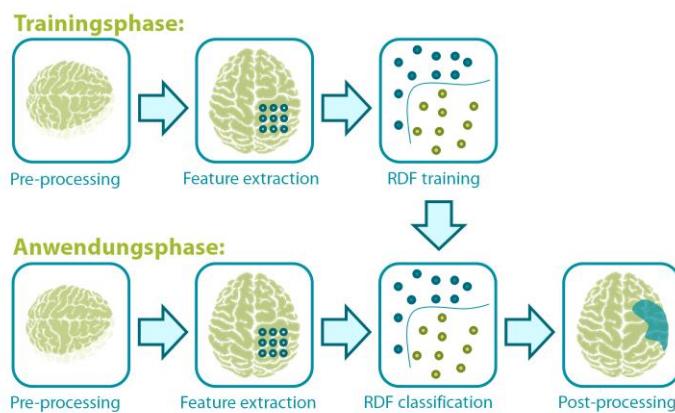
87

- Nach dem Training erhielten die Autoren diese Ergebnisse auf ihrer Evaluationsmenge
- Verschiedene Trainingsparameter wurden verglichen, der Pfeil zeigt auf ein ausgewogenes Ergebnis zwischen Falsch-Positiven und Falsch-Negativen
- Was ich an der Arbeit interessant finde ist der durchgeföhrte Vergleich mit anderen Klassifizierungsmethoden. Beide Versionen der SVM schneiden deutlich besser ab.

Anwendungsbeispiele: RDF

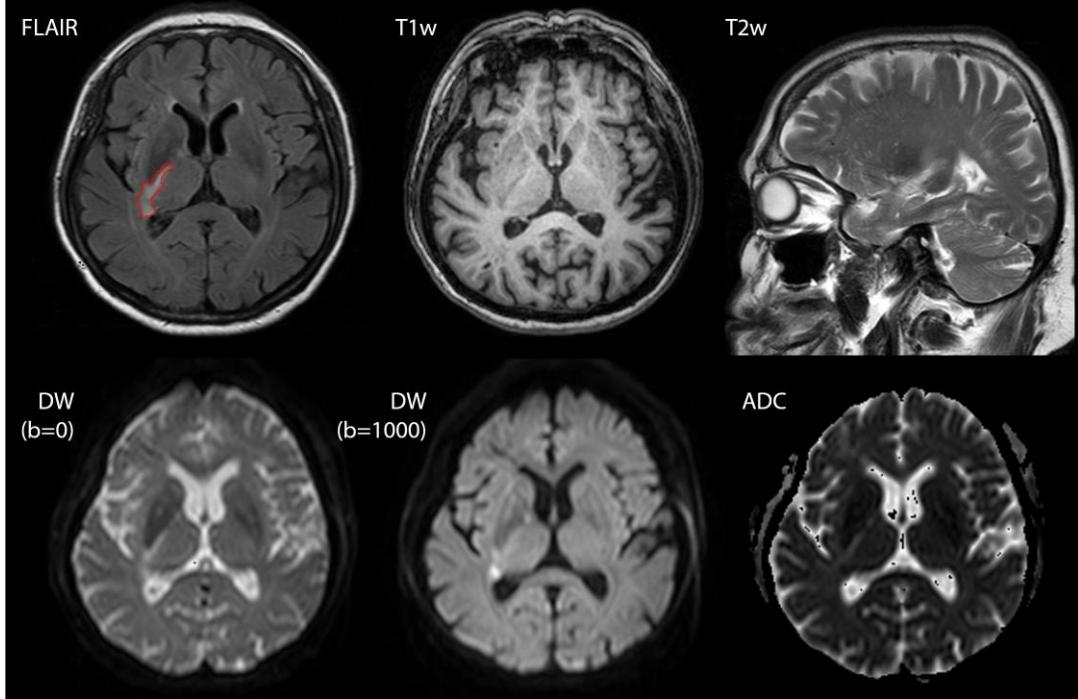
Maier et al., „Segmentierung von ischämischen Schlaganfall-Läsionen in multispektralen MR-Bildern mit RDFs“

Bildverarbeitung für die Medizin 2014



88

- Dieses Beispiel zeigt eine meiner eigenen Arbeiten.
- Für die Segmentierung ischämische Schlaganfallläsionen in Hirnscans habe ich einen RDF auf Trainingsdaten trainiert.
- Kommt ein neuer Fall herein, so werden eine Anzahl von Merkmalen für jeden Pixel extrahiert, diese an den RDF zur Klassifizierung übergeben und somit erhalten wir die Segmentierung.

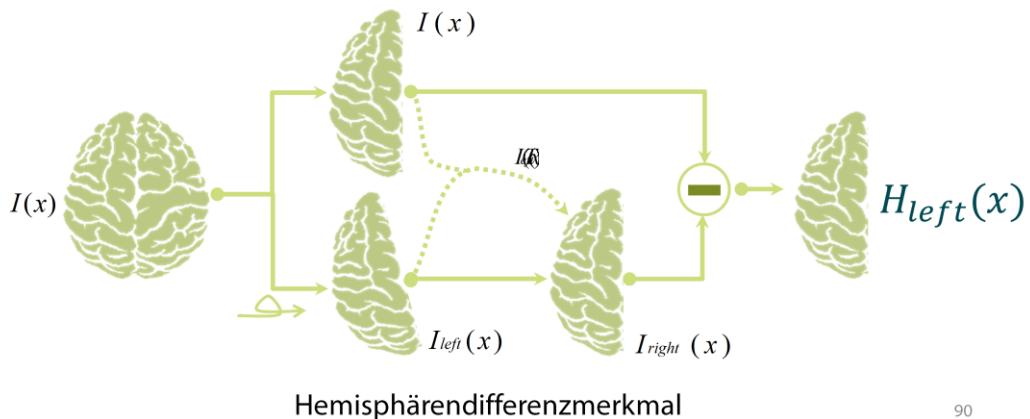


- Beispieldatensatz
- Schlaganfall-Läsion rot eingerahmt
- Sechs MR-Sequenzen = 6 Kanäle
- Der Grauwert eines Pixels im jeden Kanal stellt ein Merkmal dar = 6 Merkmale
- Weitere Merkmale werden extrahiert

Anwendungsbeispiele: RDF

Maier et al., „Segmentierung von ischämischen Schlaganfall-Läsionen in multispektralen MR-Bildern mit RDFs“

Bildverarbeitung für die Medizin 2014



Hemisphärendifferenzmerkmal

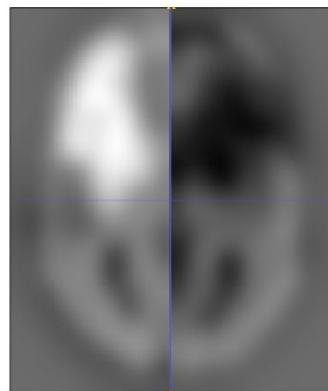
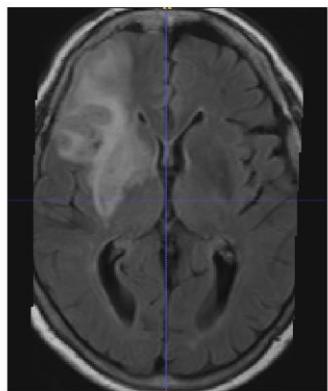
90

- Beispielsmerkmal: Hemisphärendifferenz
- Grundidee: Läsionen treten normalerweise nur in einer Hirnhemisphäre auf
- D.h. sie sind unsymmetrisch
- Wir trennen das Hirn entlang des Main Longitudinal Fissure
- Dann flippen wir eine Seite und registrieren sie auf die Andere, um kleine anatomische Unterschiede zu entfernen
- Schließlich wird noch die Differenz der Bilder gebildet

Anwendungsbeispiele: RDF

Maier et al., „Segmentierung von ischämischen Schlaganfall-Läsionen in multispektralen MR-Bildern mit RDFs“

Bildverarbeitung für die Medizin 2014

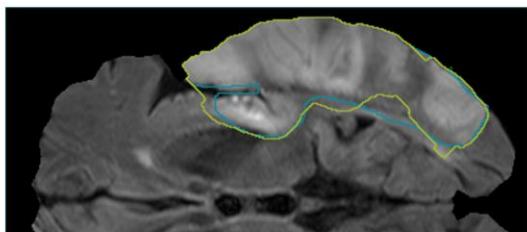


Hemisphärendifferenzmerkmal

91

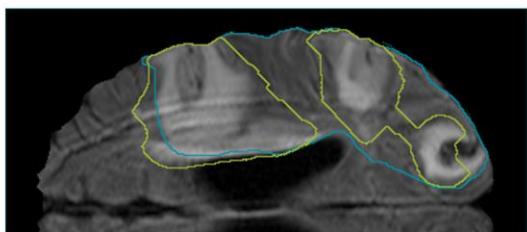
- Das sieht dann z.B. ungefähr so aus

Anwendungsbeispiele: RDF (Maier et al.)



Bester Fall

Fall Nr.	02
Dice Coeff.	0.874
Hausdorff Dist.	11.36 mm
Avg. Surface Dist.	1.55 mm



Schlechtester Fall

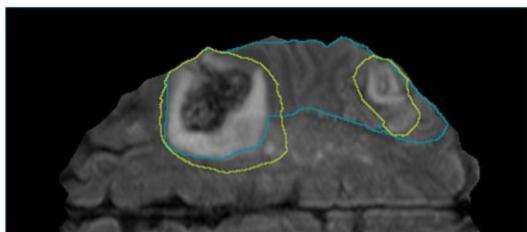
Fall Nr.	04
Dice Coeff.	0.780
Hausdorff Dist.	41.85 mm
Avg. Surface Dist.	3.57 mm

ground truth / segmentation

92

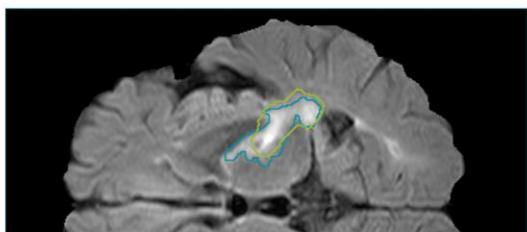
- Und hier ein paar Ergebnisbeispiele

Anwendungsbeispiele: RDF (Maier et al.)



Haemorrhage

Fall Nr.	04
Dice Coeff.	0.780
Hausdorff Dist.	41.85 mm
Avg. Surface Dist.	3.57 mm



Kleinste Läsion

Fall Nr.	03
Dice Coeff.	0.706
Hausdorff Dist.	9.92 mm
Avg. Surface Dist.	1.22 mm

ground truth / segmentation

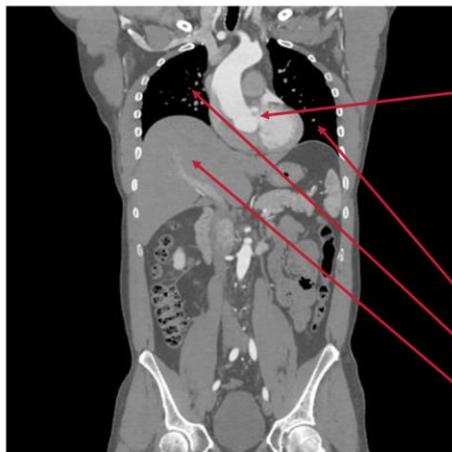
93

- Und typische Probleme die man bei realen Szenarien antrifft:
 - Im oberen Beispiel gab es eine Einblutung, die der Baum nur schwer als zu Läsion zugehörig erkennen kann (obwohl er es hier geschafft hat)
 - Im unterem Beispiel eine sehr kleine Läsion, bei der vor allem die Ausläufer nur schwer zu identifizieren sind

Anwendungsbeispiele: RDF

Criminisi et al., „Decision Forests with Long-Range Spatial Context for Organ Localization in CT Volumes“

MICCAI Workshop on Probabilistic Models for Medical Image Analysis 2009



- head
- heart
- left eye
- right eye
- l. kidney
- r. kidney
- l. Lung
- r. Lung
- liver

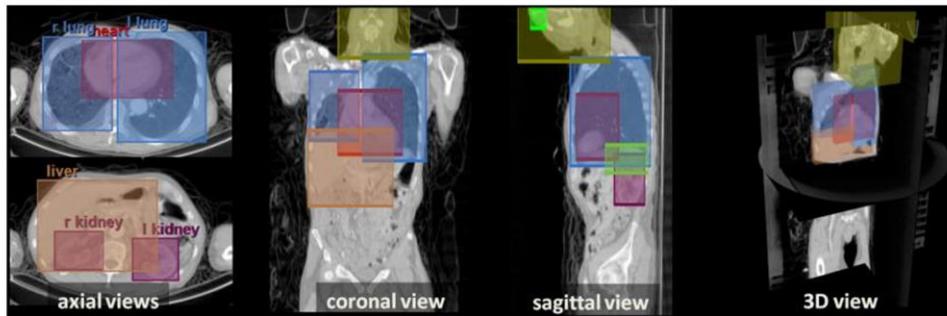
94

- Criminisi ist einer der (vielen) Eltern der RDF
- In diesem Artikel hat er eine Methode veröffentlicht, um Organ mithilfe von RDFs zu lokalisieren (keine genaue Segmentierung, nur ungefähr erkennen, wo sich das Organ befindet)
- Also eine Klassifikation der Pixel in 10 Klassen: Je eine für die Organzugehörigkeit und eine weitere für den Hintergrund
- Das Beispielsbild ist hier 2D, die Daten sind natürlich CT Volumendaten in 3D

Anwendungsbeispiele: RDF

Criminisi et al., „Decision Forests with Long-Range Spatial Context for Organ Localization in CT Volumes“

MICCAI Workshop on Probabilistic Models for Medical Image Analysis 2009



Trainingsdaten

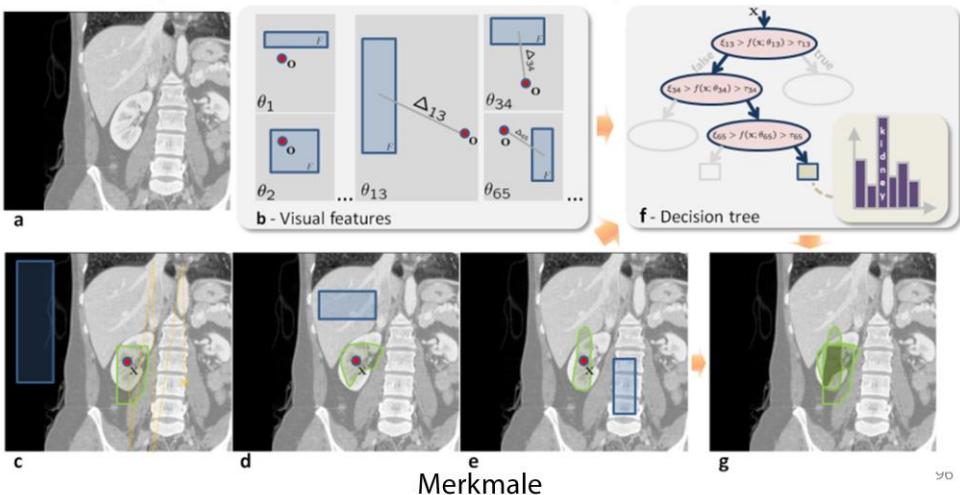
95

- So sehen ihre Trainingssets aus. Die Organe sind nur sehr grob durch farbige Boxen markiert.

Anwendungsbeispiele: RDF

Criminisi et al., „Decision Forests with Long-Range Spatial Context for Organ Localization in CT Volumes“

MICCAI Workshop on Probabilistic Models for Medical Image Analysis 2009

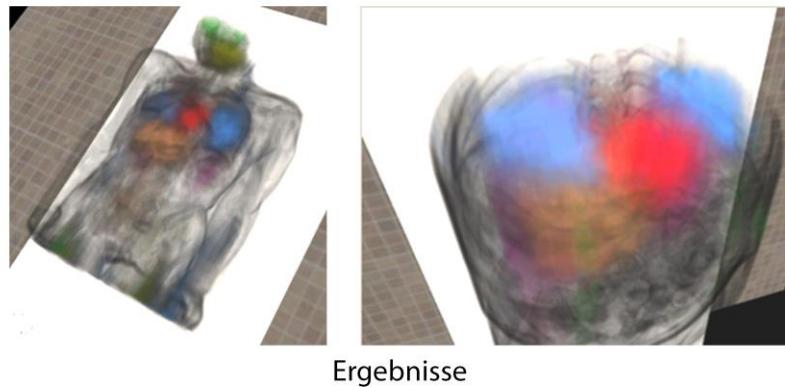


- Das interessante an der Arbeit sind die Merkmale: Die Autoren haben ausgenutzt, dass an jedem Knoten nur einen Menge von z.B. 200 Merkmalen untersucht wird und nicht die gesamte verfügbare Menge.
- Ihre Idee war nun: Warum übergeben wir dem RDF nicht eine unendliche Menge von Merkmalen als Trainingsmenge? Diese würde sich natürlich nicht vorberechnen lassen. Stattdessen werden die Merkmale erst berechnet, wenn sie an einem Knoten verglichen werden sollen => Je Knoten also 200. Sicher, das sind viele, aber doch bedeutend weniger als unendlich.
- Ihre Merkmale sind die mittleren Grauwerte von Boxen beliebiger (!) Größe in beliebiger (!) Position zum gerade betrachteten Voxel, wie in Bildteil b) dargestellt. An den Knoten bestimmt der RDF also 200 Kombinationen aus Boxgröße, Winkel und Entfernen und der zugehörige mittlere Grauwert wird erst dann berechnet.
- D.h. dem RDF stehen unendlich mögliche Kombinationen zur Verfügung => Aber nur wenige von diesen müssen realisiert werden.
- Der RDF kann durch diese Merkmale Regeln lernen: Was sich ~200 Pixel rechts von einem schwarzen Bereich befindet und ~100 Pixel unterhalb eines hellgrauen und wenige Pixel links eines eher hellen, dass wird Niere sein.
- Dies zeigen die Bildteile c) bis g).

Anwendungsbeispiele: RDF

Criminisi et al., „Decision Forests with Long-Range Spatial Context for Organ Localization in CT Volumes“

MICCAI Workshop on Probabilistic Models for Medical Image Analysis 2009



97

- Und so sehen die Organwahrscheinlichkeiten des gelernten RDF für einen neuen Fall aus. Neat, isn't it?



Quellen & Literatur

Vorlesungen von Professor Yaser Abu-Mostafa @ Caltech

- 14 – SVM
 - <https://www.youtube.com/watch?v=eHsErIPJWUU>
- 15 - Kernel Trick & Soft-Margin:
 - <http://www.youtube.com/watch?v=XUj5JbQihlU>

Christopher J.C. Burges SVM Tutorial

- <http://research.microsoft.com/pubs/67119/svmtutorial.pdf>

Criminisis Arbeiten zu RDFs

- [http://research.microsoft.com/pubs/155552/decisionForests_MSR TR 2011 114.pdf](http://research.microsoft.com/pubs/155552/decisionForests_MSR_TR_2011_114.pdf)

Einfache und machtvolle Bibliothek für Klassifizierer in Python

- <http://scikit-learn.org/>



98

- Ich empfehle vor allem die Vorlesungen, die wirklich sehr informativ und interessant gestaltet wurden
- Scikit-Learn ist eine Python-Bibliothek mit vielen Klassifizierern (u.a. SVMs und RDFs), die sehr einfach zu verwenden und mit einer guten Dokumentation ausgestattet ist



Q&A

Ende