Machine Learning
Support Vector Machines
Kernels for (Splice) Site Recognition

Soeren.Sonnenburg@first.fraunhofer.de

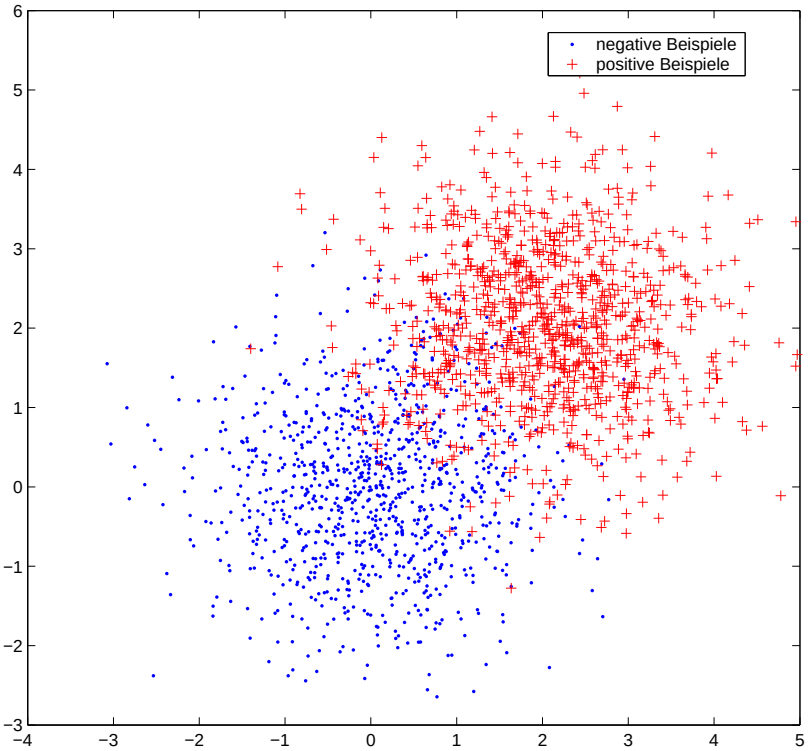
Fraunhofer Institut
Rechnerarchitektur
und Softwaretechnik

Maschinelles Lernen ... ?

- **Unüberwachtes Lernen**
  - Visualisierung
  - Dimensionsreduktion
  - Quellentrennung
  - Clustering
  - Merkmals Extraktion
- **Überwachtes Lernen**
  - K-Nächster Nachbar
  - Neuronale Netze, Perceptron
  - Support Vector Machine
  - Regression

Klassifikation ?

- Trainingsdaten $(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \{-1, +1\}$, $i = 1 \ldots N$ ($\mathcal{X}$ zB. $\mathbb{R}^2$)
- Funktion $f(\boldsymbol{x}, \boldsymbol{\alpha}) : \mathcal{X} \times \{-1, +1\}$ mit $\boldsymbol{x} \mapsto y$



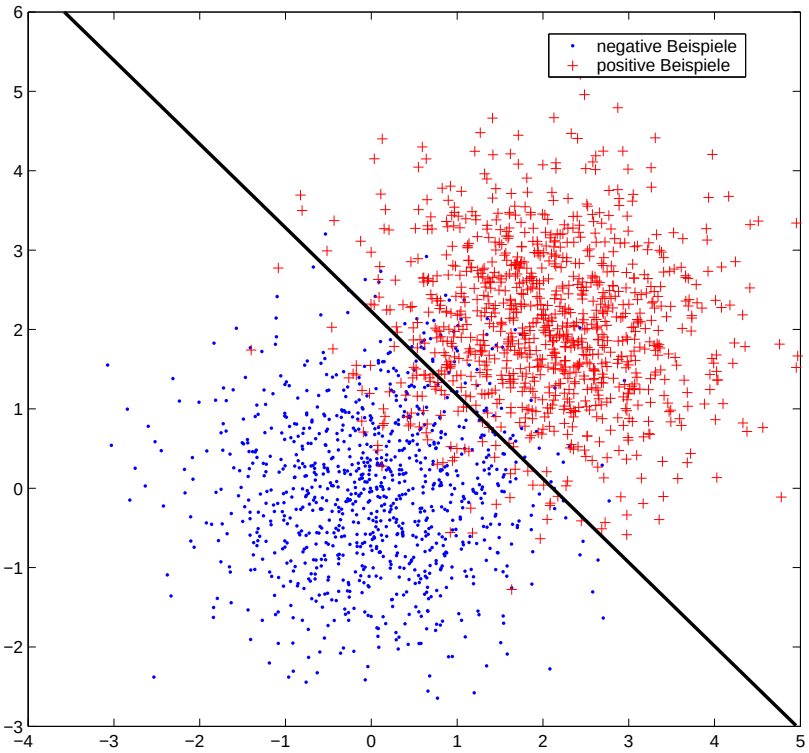
Klassifikation ?

- Trainingsdaten $(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \{-1, +1\}$, $i = 1 \ldots N$ ($\mathcal{X}$ zB. $\mathbb{R}^2$)
- Funktion $f(\boldsymbol{x}, \boldsymbol{\alpha}) : \mathcal{X} \times \{-1, +1\}$ mit $\boldsymbol{x} \mapsto y$

zB. Lineare Trennfunktion:

Parameter:
$\boldsymbol{\alpha} = (\boldsymbol{w}, b)$

Klassifikator:
$f(\boldsymbol{x}) = \text{sign}(\boldsymbol{w} \cdot \boldsymbol{x} + b)$



Klassifikator

Klassifikator

- Trainingsdaten $(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \{-1, +1\},\ i = 1 \ldots N$ unabhängig voneinander und gleichmäßig von fester Wahrscheinlichkeitsverteilung aufgenommen (i.i.d.)

- Klassifikator: Funktion $f(\boldsymbol{x}, \boldsymbol{\alpha}) : \mathcal{X} \times \{-1, +1\}$ mit $\boldsymbol{x} \mapsto y$

- Vorhersage auf **ungesehenen** Daten (von gleicher Verteilung wie Trainingsdaten)

Klassifikator ala Fisher (1936)

Annahme 2 normalverteilte Klassen,

$$i \in \{1,2\}: \; \mathcal{N}_{\boldsymbol{m}_i,\Sigma_i}(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}\,|\Sigma_i|}\mathbf{e}^{\frac{1}{2}(\boldsymbol{x}-\boldsymbol{m}_i)^\top \Sigma_i (\boldsymbol{x}-\boldsymbol{m}_i)}$$

Klassifikator:

$$\begin{aligned} f(\boldsymbol{x}) &= \operatorname{sign}\left(\mathcal{N}_{\boldsymbol{m}_1,\Sigma_1}(\boldsymbol{x}) - \mathcal{N}_{\boldsymbol{m}_2,\Sigma_2}(\boldsymbol{x})\right) \\ &= \operatorname{sign}\left(\ln \frac{\mathcal{N}_{\boldsymbol{m}_1,\Sigma_1}(\boldsymbol{x})}{\mathcal{N}_{\boldsymbol{m}_2,\Sigma_2}(\boldsymbol{x})}\right) \\ &= \operatorname{sign}\left(\frac{1}{2}(\boldsymbol{x}-\boldsymbol{m}_1)^\top \Sigma_1^{-1}(\boldsymbol{x}-\boldsymbol{m}_1) - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{m}_2)^\top \Sigma_2^{-1}(\boldsymbol{x}-\boldsymbol{m}_2) + \ln\frac{|\Sigma_2|}{|\Sigma_1|}\right) \end{aligned}$$

Klassifikator ala Fisher (1936)

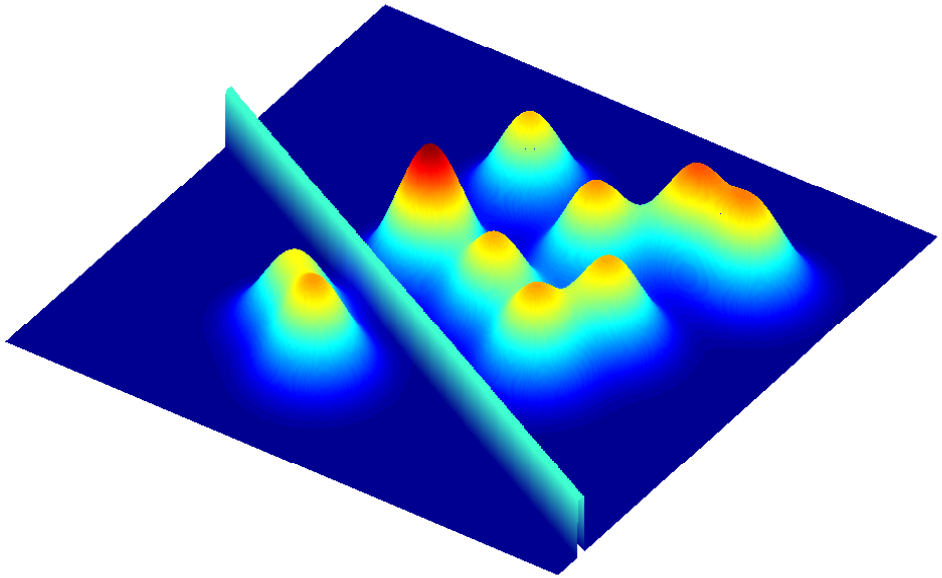
$$f_{sq}(\boldsymbol{x}) = \text{sign}\left(\frac{1}{2}(\boldsymbol{x}-\boldsymbol{m}_1)^\top\Sigma_1^{-1}(\boldsymbol{x}-\boldsymbol{m}_1) - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{m}_2)^\top\Sigma_2^{-1}(\boldsymbol{x}-\boldsymbol{m}_2) + \ln\frac{|\Sigma_2|}{|\Sigma_1|}\right)$$

$$f_{lin}(\boldsymbol{x}) = \text{sign}\left((\boldsymbol{m}_2-\boldsymbol{m}_1)^\top\Sigma^{-1}\boldsymbol{x} - \frac{1}{2}\left(\boldsymbol{m}_2^\top\Sigma^{-1}\boldsymbol{m}_2 + \boldsymbol{m}_1^\top\Sigma^{-1}\boldsymbol{m}_1\right)\right)$$

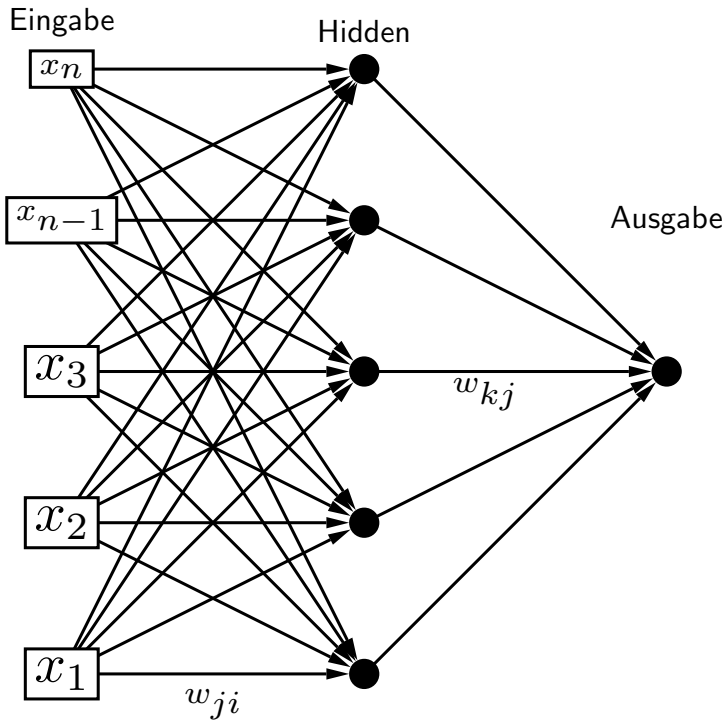
- $O(n^2)$ freie Parameter im quadratischen Fall (n Anzahl der Dimensionen)
- $O(n)$ Parameter im linearen Fall
- Fisher schlägt vor: $\Sigma = \tau\Sigma_1 + (1-\tau)\Sigma_2$ wenn $N < 10 * n^2$
- SVM nur linear

Probleme

- Normalerweise Klassenverteilungen unbekannt
- „Curse of Dimensionality“
- Nur Trennfläche für Klassifikation interessant

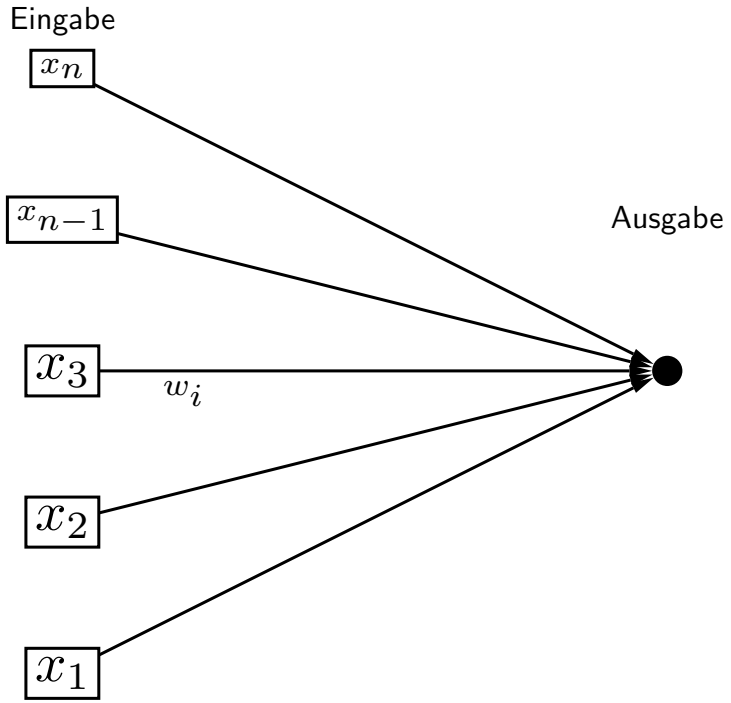


Neuronales Netz



$$f(\boldsymbol{x}) = \Phi\left(\sum_{j=1}^{n_H} w_{kj}\Phi\left(\sum_{i=1}^{N} w_{ji}x_i + w_{j0}\right) + w_{k0}\right)$$

Perceptron



Linearer Klassifikator !

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} w_i x_i + w_0 = \boldsymbol{w} \cdot \boldsymbol{x} + b$$

Probleme des Lernens

- Dichteschätzungen $\rightarrow$ „Curse of Dimensionality“
- Nur Trennfläche für Klassifikation interessant
- Lokale Minima vermeiden

Problem: Generalisierung vs. auswendig lernen

Beispiel Botaniker

Fehler

- Empirischer Fehler

$$R_{emp}(\boldsymbol{\alpha}) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\left(y_i, f(\boldsymbol{x}_i, \boldsymbol{\alpha})\right)$$

$$\mathcal{L}(y, y') = \begin{cases} 0, & \text{wenn } y = y' \\ 1, & \text{sonst} \end{cases}$$

- Erwartungswert des wahren Fehlers bei bekannter Verteilung $P(\boldsymbol{x}, y)$

$$R(\boldsymbol{\alpha}) = \int \mathcal{L}\left(y, f(\boldsymbol{x}, \boldsymbol{\alpha})\right) dP(\boldsymbol{x}, y)$$

- Bestmöglicher Klassifikator minimiert $R(\boldsymbol{\alpha})$

Bayes Klassifikator

$$
\begin{aligned}
R(\boldsymbol{\alpha}) &= \int \mathcal{L}\left(y, f(\boldsymbol{x}, \boldsymbol{\alpha})\right) dP(\boldsymbol{x}, y) \\
&= \int \mathcal{L}\left(-1, f(\boldsymbol{x}, \boldsymbol{\alpha})\right) P(y=-1, \boldsymbol{x}) d\boldsymbol{x} \\
&\quad + \int \mathcal{L}\left(+1, f(\boldsymbol{x}, \boldsymbol{\alpha})\right) P(y=+1, \boldsymbol{x}) d\boldsymbol{x}
\end{aligned}
$$

minimal für

$$
\mathcal{L}\left(-1, f(\boldsymbol{x}, \boldsymbol{\alpha})\right) = \begin{cases} 0, & P(y=-1, \boldsymbol{x}) \geq P(y=+1, \boldsymbol{x}) \\ 1, & P(y=-1, \boldsymbol{x}) < P(y=+1, \boldsymbol{x}) \end{cases}
$$

$$
\mathcal{L}\left(+1, f(\boldsymbol{x}, \boldsymbol{\alpha})\right) = \begin{cases} 0, & P(y=+1, \boldsymbol{x}) \geq P(y=-1, \boldsymbol{x}) \\ 1, & P(y=+1, \boldsymbol{x}) < P(y=-1, \boldsymbol{x}) \end{cases}
$$

FIRST

Bestmöglicher Klassifikator: Bayes Klassifikator

$$\begin{aligned} f(\boldsymbol{x}) &= \operatorname{sign}\left(P(y=+1,\boldsymbol{x}) - P(y=-1,\boldsymbol{x})\right) \\ &= \operatorname{sign}\left(P(y=+1|\boldsymbol{x})P(\boldsymbol{x}) - P(y=-1|\boldsymbol{x})P(\boldsymbol{x})\right) \\ &= \operatorname{sign}\left(P(y=+1|\boldsymbol{x}) - P(y=-1|\boldsymbol{x})\right) \end{aligned}$$

obere Schranke für Fehler

- Abschätzung des Erwartungswertes des Fehlers $0 \leq \eta \leq 1$ beliebig, mit Wahrscheinlichkeit $1 - \eta$ hält diese Ungleichung (Vapnik,1995)

$$R(\boldsymbol{\alpha}) \leq R_{emp}(\boldsymbol{\alpha}) + \underbrace{\sqrt{\left(\frac{VC(\log\frac{2N}{VC}+1) - \log\frac{\eta}{4}}{N}\right)}}_{\text{VC-Sicherheit}}$$

Vapnik Chervonenkis (VC) Dimension

Definition

- VC ist die Vapnik Chervonenkis Dimension, ein Maß für die Komplexität einer Klasse von Funktionen $\{f(\boldsymbol{x}, \boldsymbol{\alpha})\}$

- Maximale Anzahl von Punkten beliebiger Färbung, die günstig angeordnet durch mindestens eine Funktion aus dieser Klasse auseinandergehalten werden kann.
  Hier: Funktionen $\{f(\boldsymbol{x}, \boldsymbol{\alpha})\} \in \{-1, 1\}$, daher 2^N verschiedene Färbungen bei N Punkten.

- Separierung durch Hyperebenen mit Ausrichtung; $\mathbb{R}^n \rightarrow n + 1$ Punkte, Beispiel $\mathbb{R}^2$

- Beispiel: k-Nächster Nachbarschafts-klassifikator für $k = 1$ hat $R_{emp} = 0$ und VC Dimension unendlich. Abschätzung gibt keine Information, dennoch Klassifikator gut.

Strukturelle Risiko Minimierung (SRM)

- VC-Sicherheit nur von Klasse von Funktionen abhängig, **aber** empirisches/tatsächliches Risiko von der genauen Funktion abhängig.
- Minimierung durch Unterteilung der Klasse von Funktionen in verschachtelte Unterklassen für die wiederum VC bestimmt/abgeschätzt wird.
- SRM sucht die Unterklasse, die kleinste obere Schranke zum tatsächlichen Risiko ist, indem in jeder Unterklasse ein Klassifikator mit möglichst geringem empirischen Risiko bestimmt wird und letztendlich die mit kleinster Summe aus empirischem Risiko und VC Sicherheit genommen wird.

Occams Razor, KiSS

SVM

Der separierbare Fall

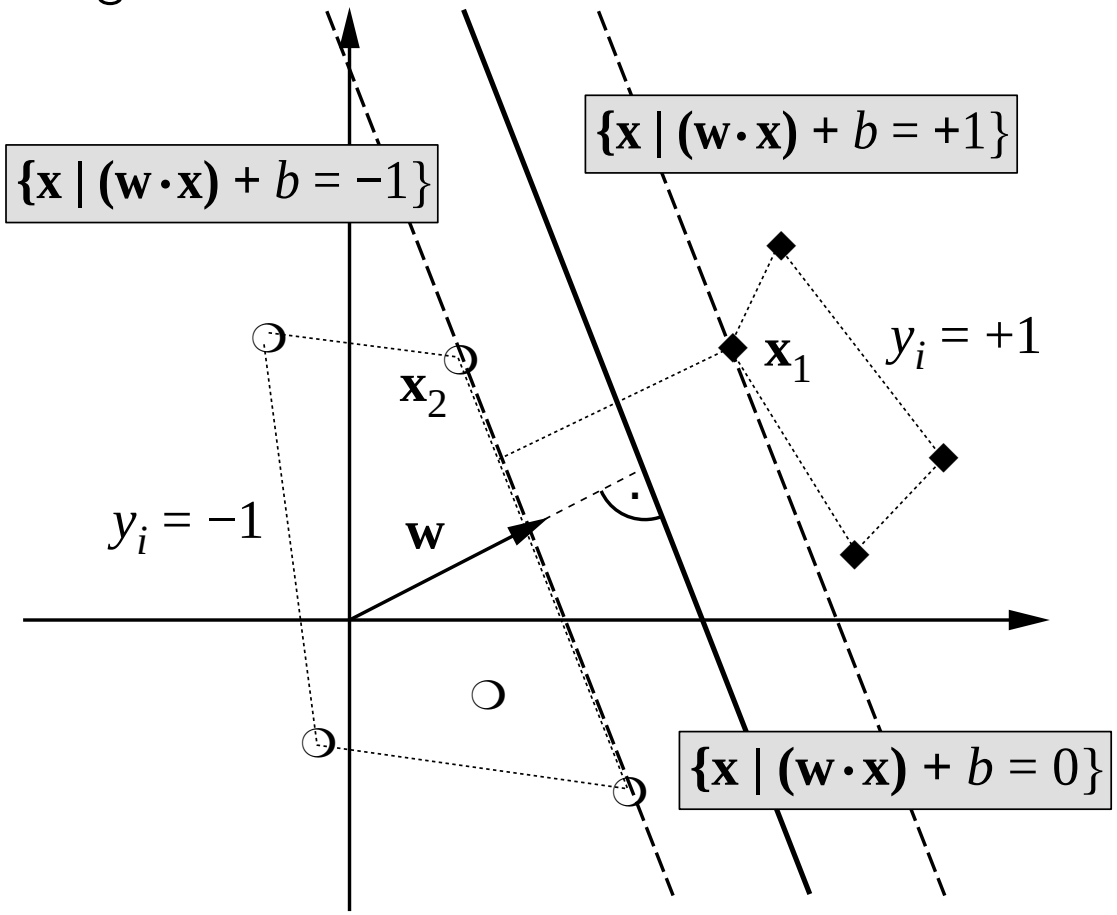
- Trainings Daten $(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \{-1, +1\}$ i.i.d.
- Support Vector Machine lernt *linearen* Klassifikator $f(\boldsymbol{x}, \boldsymbol{\alpha})$, $\boldsymbol{\alpha} = (\boldsymbol{w}, b)$.

$$f(\boldsymbol{x}) = \operatorname{sign}(\boldsymbol{w} \cdot \boldsymbol{x} + b)$$

- welcher Lineare Klassifikator „optimal“ ?
- der mit größtem „Margin“ (störungsunempfindlich in Daten und Parametern)
- globales Optimum, Komplexität nur von Anzahl der Supportvektoren nicht der Dimension des Problems abhängig

SVM

Hyperebene beschrieben durch $\boldsymbol{w}\cdot\boldsymbol{x}+b=0$ separiert unsere Trainingsdaten in positive/negative Exemplare mit $f(\boldsymbol{x}) = \operatorname{sign}(\boldsymbol{w}\cdot\boldsymbol{x}+b)$ als Entscheidungsfunktion. Problem Bestimmung der optimalen Hyperebene mit maximalem Margin:



Note:

$$(\mathbf{w}\cdot\mathbf{x}_1) + b = +1$$
$$(\mathbf{w}\cdot\mathbf{x}_2) + b = -1$$

$$\Rightarrow \quad (\mathbf{w}\cdot(\mathbf{x}_1-\mathbf{x}_2)) = 2$$

$$\Rightarrow \left(\frac{\mathbf{w}}{\|\mathbf{w}\|}\cdot(\mathbf{x}_1-\mathbf{x}_2)\right) = \frac{2}{\|\mathbf{w}\|}$$

SVM

- Senkrechter Abstand der Hyperebenen: $\frac{2}{||\boldsymbol{w}||}$ Margin
- $\Rightarrow$ maximal für minimales $||\boldsymbol{w}||$
- Bedingungen: Hyperebenen parrallel zur Entscheidungsfläche und $\boldsymbol{x}_i$ auf der „richtigen“ Seite

$$\boldsymbol{x}_i \cdot \boldsymbol{w} + b \geq +1 \text{ for } y_i = +1$$
$$\boldsymbol{x}_i \cdot \boldsymbol{w} + b \leq -1 \text{ for } y_i = -1$$

$$\Rightarrow y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) \geq +1 \text{ for } y_i = \pm 1.$$

SVM

Zu lösen:

minimiere $\frac{1}{2}\|\boldsymbol{w}\|^2$

bei Einhaltung von $y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - 1 \geq 0$ for $y_i = \pm 1,\ i = 1 \ldots N$

$\Rightarrow$ *Standard* quadratisches Optimierungsproblem

- positive Lagrange Multiplikatoren: $\alpha_i,\ i = 1, \ldots, N$. für jede Ungleichung einen. werden mit der jeweiligen Ungleichung multipliziert und von der gegenläufigen Bedingung subtrahiert.

$$L_p \equiv \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{N} \alpha_i y_i (\boldsymbol{x}_i \cdot \boldsymbol{w} + b) + \sum_{i=1}^{N} \alpha_i$$

SVM

- $\mathfrak{C}_1$: Minimierung von L_p bzgl. $\boldsymbol{w}$, b unter Berücksichtigung des Verschwindens aller Ableitungen nach α_i unter Einhaltung von $\alpha_i \geq 0$. Das ist ein konvexes quadratisches Optimierungsproblem daher äquivalente Lösung des „Dualen Problems."

- Das Duale Optimierungsproblem
  $\mathfrak{C}_2$: Maximierung von L_p unter Berücksichtigung des Verschwindens der Ableitungen von L_p nach $\boldsymbol{w}$, b und unter Einhaltung von $\alpha_i \geq 0$. „Wolf Dual" mit Eigenschaft Maximum von L_p mit Bedingungen $\mathfrak{C}_2$ hat selbe Lösungen $\boldsymbol{w}$, b, $\boldsymbol{\alpha}$ wie das Minimum unter den Bedingungen $\mathfrak{C}_1$.

SVM

$$L_p \equiv \tfrac{1}{2}\left\|\boldsymbol{w}\right\|^2 - \sum_{i=1}^{N} \alpha_i y_i (\boldsymbol{x}_i \cdot \boldsymbol{w} + b) + \sum_{i=1}^{N} \alpha_i$$

$$\frac{dL_p}{d\boldsymbol{w}} = 0 = \boldsymbol{w} - \sum_i \alpha_i y_i \boldsymbol{x}_i$$

$$\frac{dL_p}{db} = 0 = \sum_i \alpha_i y_i$$

einsetzen:

$$\begin{aligned} L_D &= \frac{1}{2}\left|\sum_i \alpha_i y_i \boldsymbol{x}_i\right|^2 - \sum_i \alpha_i y_i \boldsymbol{x}_i \cdot \sum_j \alpha_j y_j \boldsymbol{x}_j + \sum_i \alpha_i \\ &= \frac{1}{2}\sum_i \alpha_i y_i \boldsymbol{x}_i \sum_j \alpha_j y_j \boldsymbol{x}_j - \sum_i \alpha_i y_i \boldsymbol{x}_i \sum_j \alpha_j y_j \boldsymbol{x}_j + \sum_i \alpha_i \\ &= \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j \end{aligned}$$

SVM

The Karush-Kuhn-Tucker komplementär Bedingung sagt:

$$\alpha_i(y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - 1) = 0 \qquad i = 1, \ldots, N$$

Die Lösung ist nun:

$$\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i$$

$$b = \frac{1}{y_k} - \boldsymbol{x}_k \cdot \boldsymbol{w} = y_k - \boldsymbol{x}_k \cdot \left(\sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i \right)$$

Entscheidungsfunktion:

$$f(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i \cdot \boldsymbol{x} + b \right) = \operatorname{sign}\left(\left(\sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i \cdot (\boldsymbol{x} - \boldsymbol{x}_k) \right) + \frac{1}{y_k} \right).$$

Der nichtseparierbare Fall

- Einführung von „Schlupfvariablen“ ξ_i

$$\boldsymbol{x}_i \cdot \boldsymbol{w} + b \geq +1 - \xi_i, \text{ for } y_i = +1$$
$$\boldsymbol{x}_i \cdot \boldsymbol{w} + b \leq -1 + \xi_i, \text{ for } y_i = -1$$
$$\xi_i \geq 0 \qquad \forall i.$$

  ($\sum_i \xi_i$ obere Schranke für Trainingsfehler)

- Zielfunktion nun:

  Minimiere $\frac{||\boldsymbol{w}||^2}{2} + C\,(\sum_i \xi_i)$, mit C Straffaktor (je größer desto mehr wird bestraft)

- „Wolf Dual“ Maximiere:

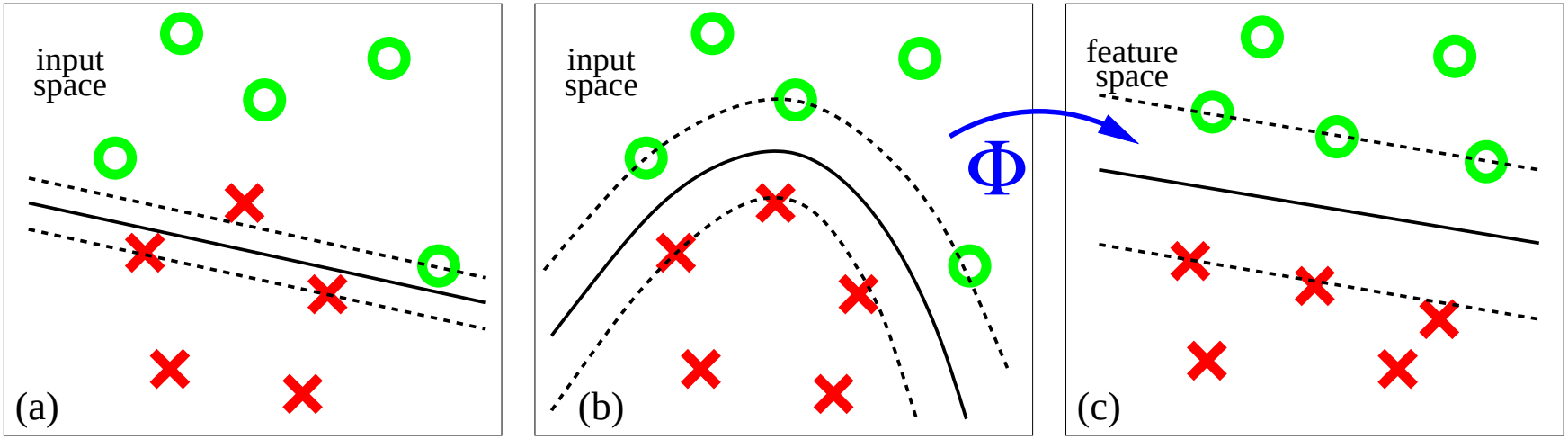
$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j$$

  unter Beachtung von $0 \leq \alpha_i \leq C$ und $\sum_i \alpha_i y_i = 0$.

- Lösung genauso (einziger Unterschied obere Schranke C)

Support vektor Kerne

- bisher nur lineare Lösung in Merkmalsräumen, aber Lineare Trennung für viele Probleme nicht möglich
- $\rightarrow$ nichtlineare Transformation $\Phi : \mathcal{X} \rightarrow \mathcal{F}$, dort wieder linear Lösung suchen
- nur Skalarprodukte $\Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{x'})$ benötigt und nicht explizit $\Phi(\boldsymbol{x})$
- Kernel Trick: $\mathrm{k}(\boldsymbol{x}, \boldsymbol{x'}) = (\Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{x'}))$



Support Vektor Kerne

- $K(\boldsymbol{x}, \boldsymbol{x}') = (\Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{x}'))$ in Duales Optimierungsproblem einsetzen:

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

- Entscheidungsfunktion wird:

$$\begin{aligned} f(\boldsymbol{x}) &= \operatorname{sign}\left(\sum_{i=1}^{N} \alpha_i y_i \left(\Phi(\boldsymbol{x})_i \cdot \Phi(\boldsymbol{x})\right) + b \right) \\ &= \operatorname{sign}\left(\sum_{i=1}^{N} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b \right) \end{aligned}$$

Kerne - Beispiele

- Polynomieller Kernel

$$\begin{aligned} \mathrm{k}(\boldsymbol{x}, \boldsymbol{x'}) &:= (\boldsymbol{x} \cdot \boldsymbol{x'})^d \\ &= \left(\sum_{i=1}^{N} x_i x'_i \right)^d \\ &= \sum_{i_1,\ldots,i_d \in \{1\ldots N\}} x_{i_1} \cdot \ldots \cdot x_{i_d} x'_{i_1} \cdot \ldots \cdot x'_{i_d} \\ &= (C_d(\boldsymbol{x}) \cdot C_d(\boldsymbol{x'})) \end{aligned}$$

$C_d(\boldsymbol{x})$ ordnet $\boldsymbol{x}$ alle bis zum Grad d möglichen Monome (nach Grad geordnet) zu Bsp: $C_2((x_1, x_2)) = (x_1^2, x_2^2, x_1 x_2, x_2 x_1)$

- Gauss RBF kernel:

$$k(\boldsymbol{x}, \boldsymbol{x}') := \mathbf{e}^{-\left(\frac{\left\| \boldsymbol{x} - \boldsymbol{x}' \right\|^2}{\sigma^2} \right)}$$

Klassifikator (SVM) Training

Standard Fall

- Aufteilung der Daten in Training,Validierung,Test
- Auswahl einer Menge geeigneter Kerne (oder neu designen)
- SVM auf Trainingsdatensatz trainieren und auf Validierungsdatensatz evaluieren. C und Kern (und Kernparameter) so einstellen, dass Validierungsfehler minimal.
- tatsächlichen Fehler auf den Testdaten schätzen

Wenig Daten $\Rightarrow$ Kreuzvalidierung

- Aufteilung der Daten in s Teile

- $s-1$ Teile für Training, den anderen Teil aufsplitten in Validierung und Test
- s mal trainieren (Extremfall $s = N - 1$)
- Auswahl einer Menge geeigneter Kerne (oder neu designen)
- SVM auf Trainingsdatensatz trainieren und auf Validierungsdatensatz evaluieren. C und Kern (und Kernparameter) so einstellen, dass Validierungsfehler minimal.
- tatsächlichen Fehler auf den Testdaten schätzen und mit Standardabweichung angeben

Vergleich von Klassifikatoren

- Test Fehler
- Fehler für bestimmten Wert von falsch negativen auf dem Receiver Operating Characteristic (ROC) Graphen
- ROC Score (Fläche unter der ROC)

Konfusions Matrix

	Klassifikator Vorhersagen		
wahre Label	tp	fn	+
	fp	tn	−
	+	−	

- Sensitivität

$$sensitivity = \frac{tp}{tp + fn}$$

- Spezifizität (Prezision)

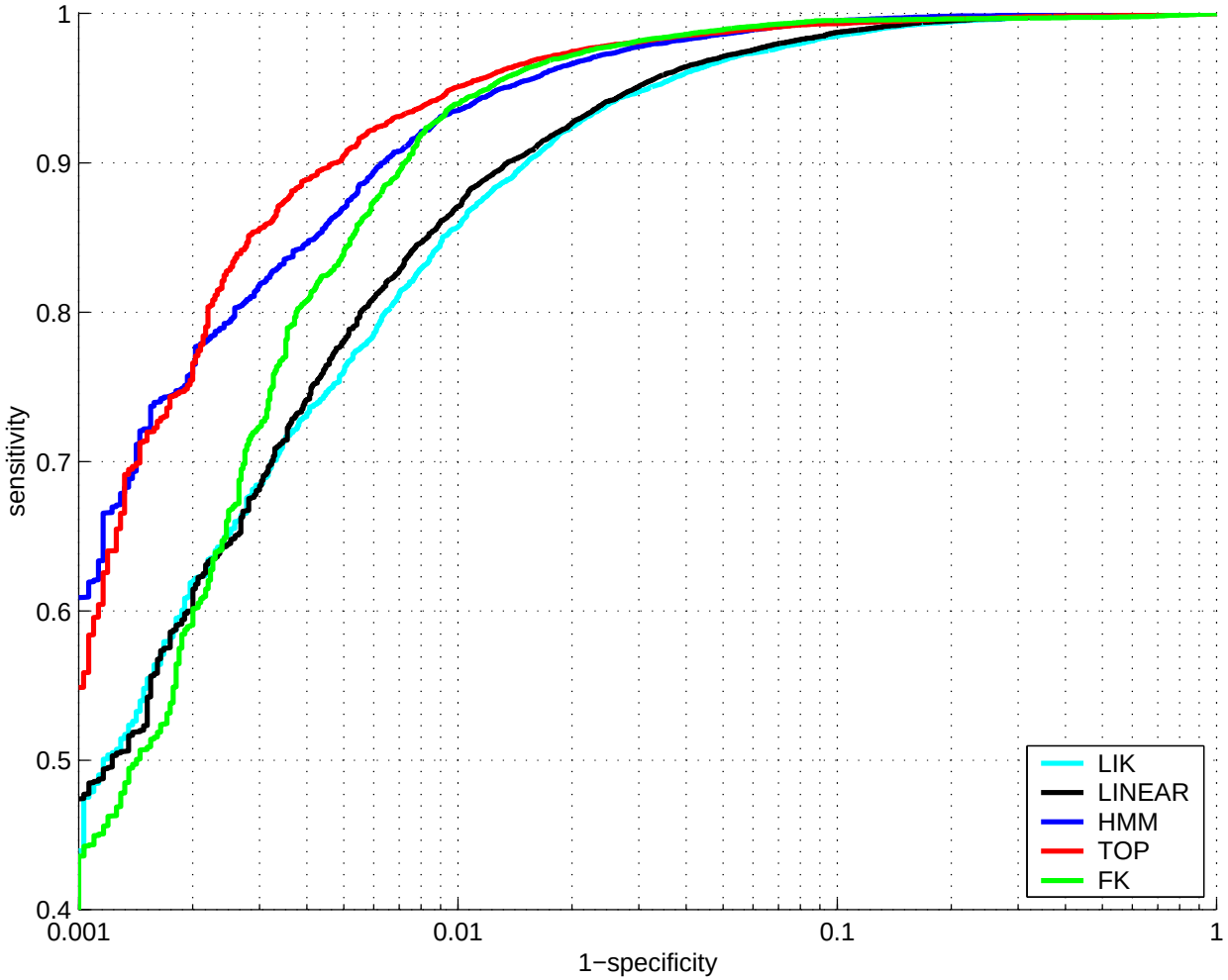
$$specificity = \frac{tn}{fp + tn}$$

ROC

Receiver Operating Characteristic

$$sens = \frac{tp}{tp + fn}$$

$$spec = \frac{tn}{fp + tn}$$

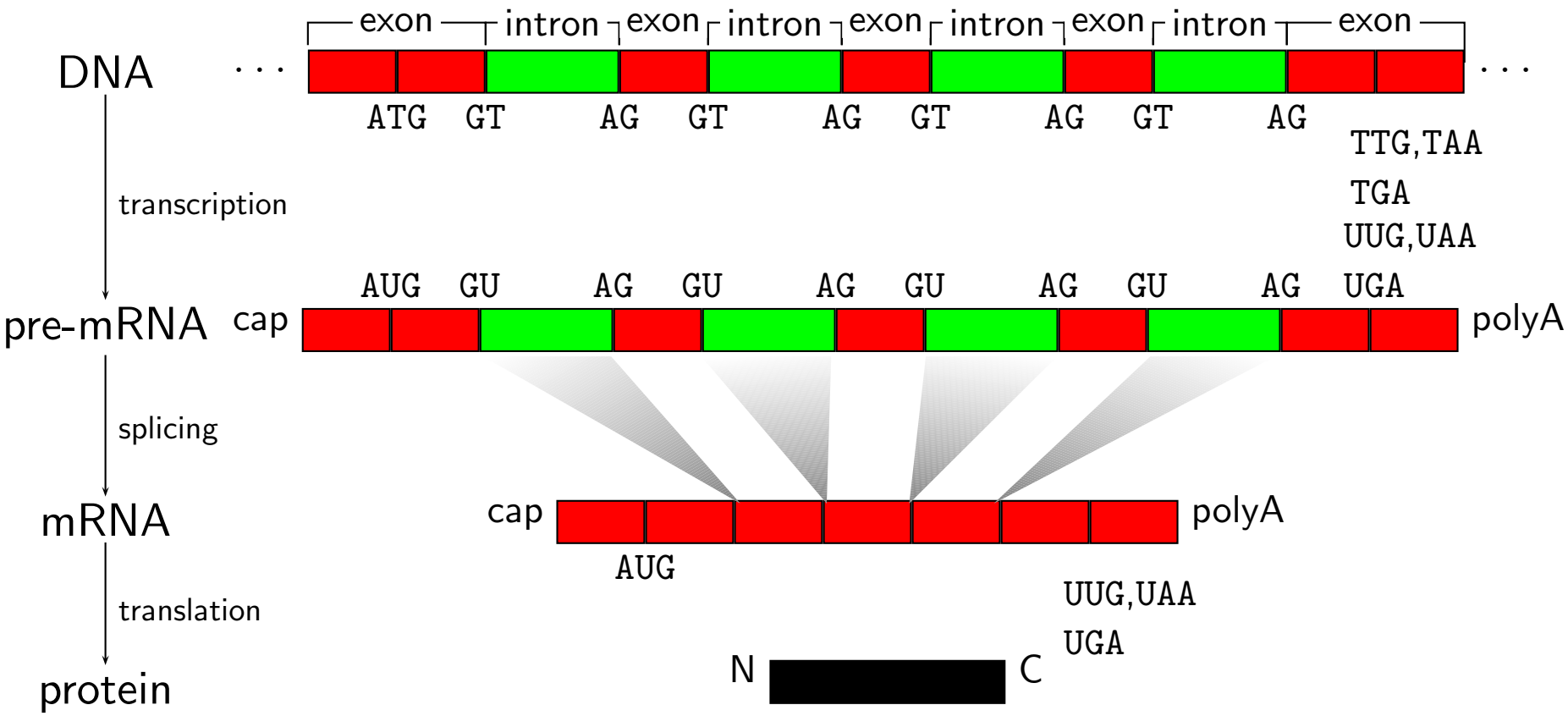


Spleissstellen-Erkennung

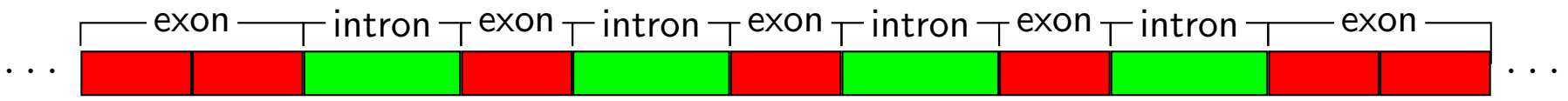
- SVM zur kanonischen Spleißstellenerkennung als Beispiel
- Vorwissen in Kern einbauen
- Was sind nützliche Eigenschaften
- Daten Extraktion
- Formulierung als 2 Klassen Klassifikationsproblem
- Welche Kerne machen Sinn ? Neudesign ?

Biologischer Hintergrund

Spleißstellen sind Übergangsstellen auf der DNA zwischen
- exons (Protein kodierend)
- introns (nicht kodierend)



Fakten über Spleissstellen



- Exons sehr kurz ($100 - 200$ bp); Introns können sehr lang sein (> 1 kbp)
- Spleißen findet im *Zellkern* statt.
- Der Spleißapparat ("**Spliceosome**") ist nicht Gewebespezifisch
- Spleißmechanismus sehr ähnlich in Eukaryoten
- Experimente zeigen, daß jede 5' Stelle mit jeder 3' Stelle verknüpft werden kann

Spleißmechanismus verwendet *locale* Information

Wieso Spleissstellen Erkennung wichtig?

- erlaubt *akkurate* Vorhersage von mRNA and somit von Proteinen aus DNA

  $\Rightarrow$ wichtiger Schritt bei der Analyse des Genoms

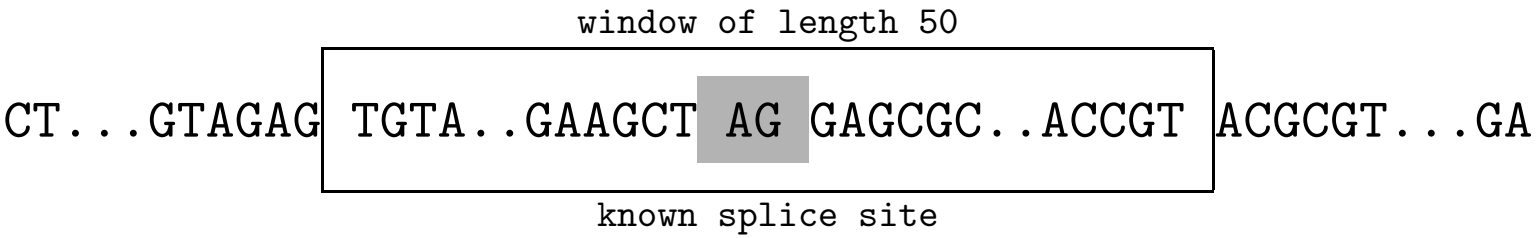
- Spleißstellen können sehr gut erkannt werden

  $\Rightarrow$ wichtiger und exakter 'Marker' um Gene zu finden

Standard: Alignment to Datenbank Einträgen

Ziel: Spleißen mit Hilfe von ML verbessern

2-Klassen Klassifikations Problem



- nur kanonische Spleißstellen (consensus AG,GT, 98%)
- wahre Spleißstellen: festes Fenster um die Spleißstelle
- falsche: erzeugt, durch verschieben des Fensters ± 25 bp

```
AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAACAAATTTTTAGCATTACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAGTTCACTAACACATCCGTCTGTGCC
TTAATTTCACTTCCACATACTTCCAGATCATCAATCTCCAAAACCAACAC
TTGTTTTAATATTCAATTTTTTACAGTAAGTTGCCAATTCAATGTTCCAC
TACCTAATTATGAAATTAAAATTCAGTGTGCTGATGGAAACGGAGAAGTC
```

Download at http://mlg.anu.edu.au/~raetsch/splice

Nochmal: Eigenschaften von Spleissstellen

- Konservierte Sequenz
- Positionsabhängig
- Mutationen
- Einfügungen
- Löschungen
- Verschoben

Support Vector Machines

- Sequenzen $\boldsymbol{x}_i \in \mathcal{X}$ $(i = 1, \ldots, N)$ with respective labels y_i
- SVM Klassifikator (im Prinzip: Perceptron im Kernel Feature Space):

$$f(\boldsymbol{x}) = \text{sign}\left(\sum_{i=1}^{N} y_i \alpha_i \text{k}(\boldsymbol{x}, \boldsymbol{x}_i) + b\right)$$

- lernt Parameter $\boldsymbol{\alpha}$ durch Lösung des quadratischen Optimierungsproblems:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \, \text{k}(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

subject to $\alpha_i \in [0, C]$, $i = 1, \ldots, N$, $\sum_{i=1}^{N} \alpha_i y_i = 0$.

Lösung hat keine lokalen Minima

Kerne - linear/polynomieller Kern

Polynomieller Kern vom Grad d:

$$\mathrm{k}_{\mathrm{POLY}}(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}, \boldsymbol{x}')^d$$

- nicht direkt anwendbar
- Ansatz, Eingabe Raum in binär Raum transformieren oder equivalent matches zählen

$$\mathrm{k}_{\mathrm{POLY}}(\boldsymbol{x}, \boldsymbol{x}') = \left(\sum_{p=1}^{N} \mathsf{I}_p(\boldsymbol{x}, \boldsymbol{x}') \right)^d$$

$\Rightarrow$ Merkmalsraum alle Monome d-ter Ordnung, d.h.

$$x_{i_1} \cdot x_{i_2} \cdot \ldots \cdot x_{i_d},\ i_{1\ldots d} \in 1 \ldots N$$

$\Rightarrow$ sehr hochdimensional, *nicht robust*

Kernels - locality improved kernel

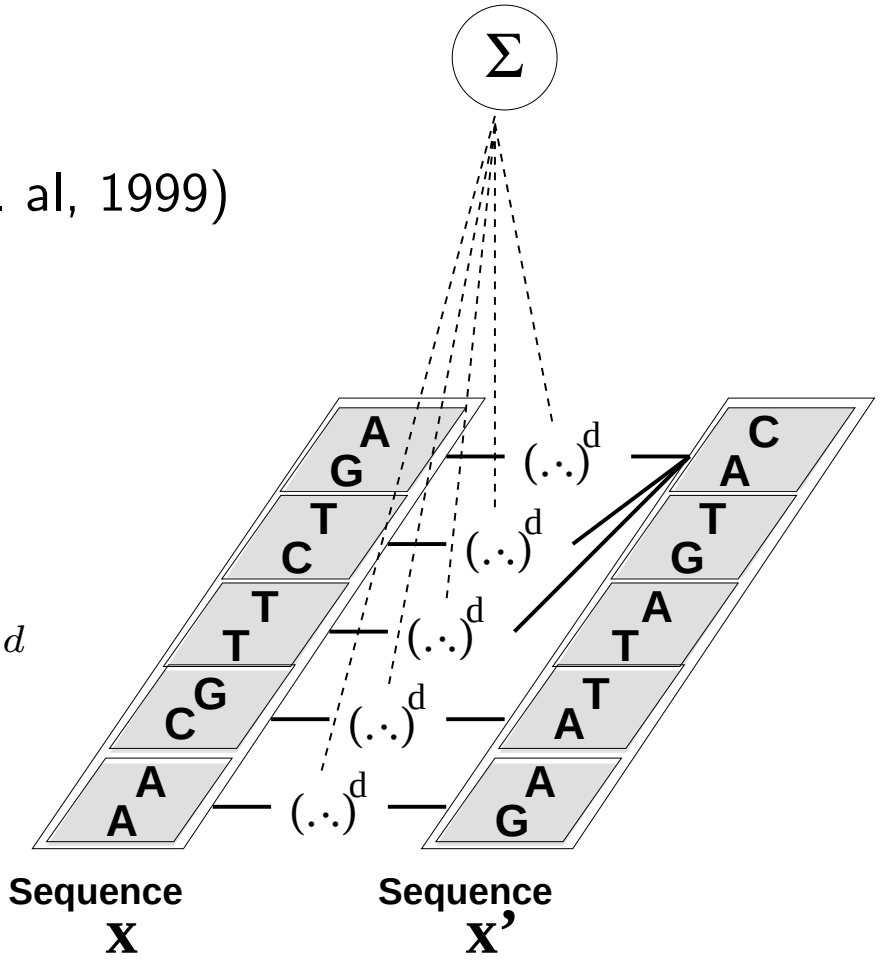
Locality Improved Kernel (Zien et. al, 1999)

- im Prinzip polynomieller Kern der locale Information betont

$$\mathrm{k_{LI}}(\boldsymbol{x}, \boldsymbol{x'}) = \sum_{p=1}^{N} \mathrm{win}_p(\boldsymbol{x}, \boldsymbol{x'})$$

$$\mathrm{win}_p(\boldsymbol{x}, \boldsymbol{x'}) = \left(\sum_{j=-l}^{+l} p_j \mathrm{I}_{p+j}(\boldsymbol{x}, \boldsymbol{x'}) \right)^d$$

$$\mathrm{I}_i(x, x') = \begin{cases} 1, & x_i = x'_i \\ 0, & \text{otherwise} \end{cases}$$



$\Rightarrow$ zählt letztendlich Matches von $2l+1-$ Tupeln

Weighted Degree Kernel

- zählt Matches zwischen zwei Sequenzen $\boldsymbol{x}$ and $\boldsymbol{x}'$ zwischen den Worten $\boldsymbol{u}_{\omega,i}(\boldsymbol{x})$ und $\boldsymbol{u}_{\omega,i}(\boldsymbol{x}')$
- $\boldsymbol{u}_{\omega,i}(\boldsymbol{x}) = x_i x_{i+1} \dots x_{i+\omega-1}$ für alle i und $1 \leq \omega \leq d$.
- ω ist die Ordung (oder Länge des zu vergleichenden Wortes).

Weighted Degree Kernel

$$\mathrm{k}_{WD}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{\omega=1}^{d} w_\omega \sum_{i=1}^{N-d} \mathrm{I}(\boldsymbol{u}_{\omega,i}(\boldsymbol{x}) = \boldsymbol{u}_{\omega,i}(\boldsymbol{x}'))$$

- w_ω Gewicht für Match der Länge ω
- funktioniert auch mit Mismatches können (geringere Gewichte für nicht perfekte Matches)

Weighted Degree Position Kernel

$$\mathrm{k}_{WDpos}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{\omega=1}^{d} w_\omega \sum_{i=1}^{N-d-\mathrm{shift}_i} \sum_{j=1}^{\mathrm{shift}_i} \mathrm{I}(\boldsymbol{u}_{\omega,i+j}(\boldsymbol{x}) = \boldsymbol{u}_{\omega,i+j}(\boldsymbol{x}'))$$

Kern kann folgende Eigenschaften ausnutzen

- Konservierte Sequenz
- Positionsabhängig
- Mutationen
- Einfügungen
- Löschungen
- Verschoben

Spectrum Kernel

Spectrum Kernel (Leslie et. al, 2002):

$$\mathrm{k}_{spec} = s(\boldsymbol{x})^\top s(\boldsymbol{x})$$

Skalarprodukt aus

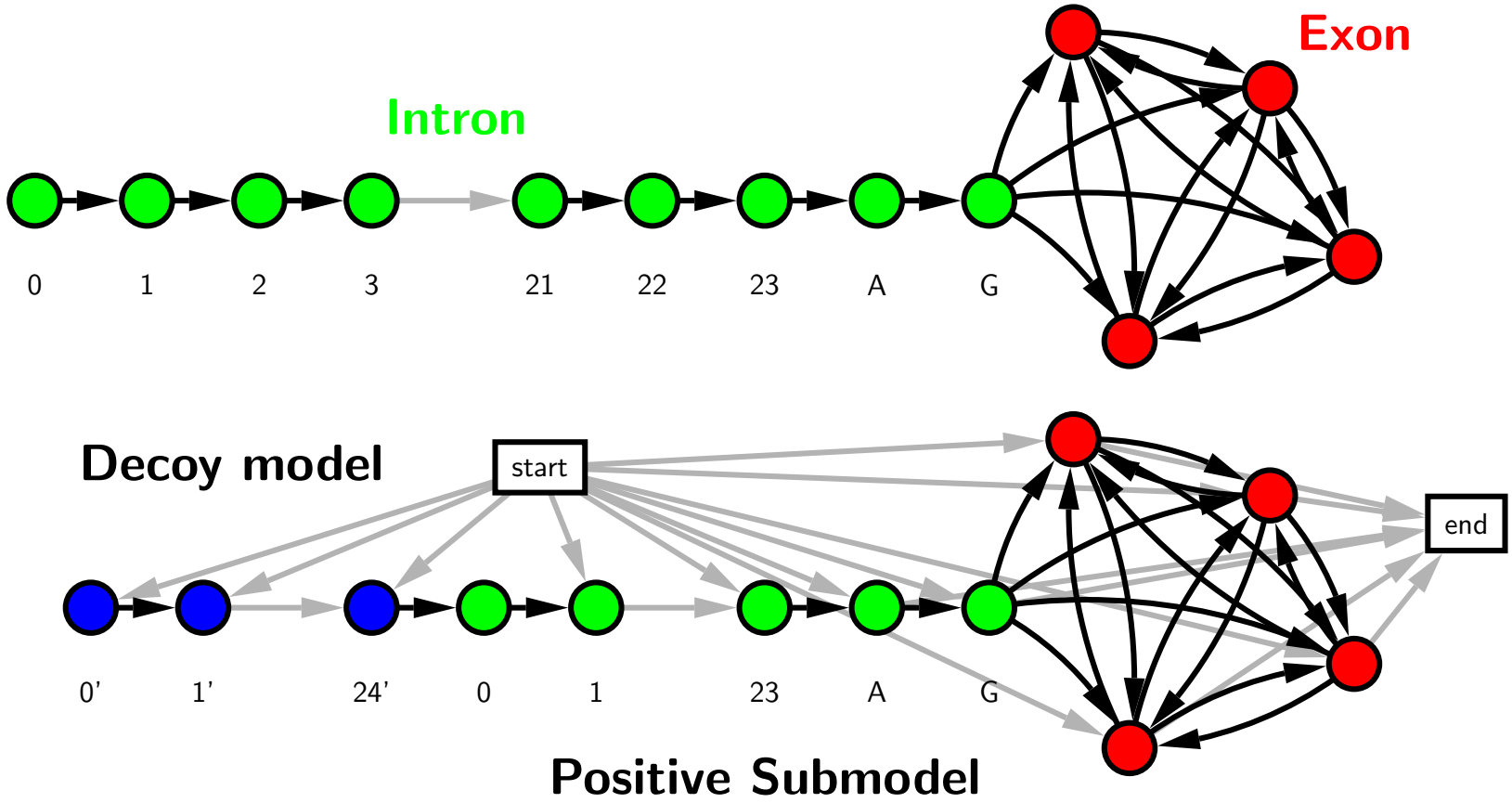
$$s(\boldsymbol{x}) = (\#A(\boldsymbol{x}))_{A \in \Sigma^k}$$

wobei $\#A(\boldsymbol{x}) =$ Häufigkeit des Auftretens von A in $\boldsymbol{x}$.

- zählt Matches von k-Tupeln
- keine positionsabhängige Information

Generative Modelle

Verwenden generatives Model, z.B. designen ein HMM



(oben) positive Acceptor Model, (unten) negative Acceptor model

Kerne

- Kernel von generativen Modellen
  - vergleicht Objekte durch ein generative Model $\Pr(\boldsymbol{x}|\Theta)$
  - verwendet probabilistisches Model zum discriminativen Training

- Fisher Kernel (Jaakkola and Haussler, 1998)

$$\begin{aligned} \mathrm{k}_{\mathrm{FK}}(\boldsymbol{x},\boldsymbol{x}') &= \boldsymbol{s}(\boldsymbol{x},\hat{\boldsymbol{\theta}})^{\top} Z^{-1}(\hat{\boldsymbol{\theta}}) \boldsymbol{s}(\boldsymbol{x}',\hat{\boldsymbol{\theta}}) \\ \boldsymbol{s}(\boldsymbol{x},\hat{\boldsymbol{\theta}}) &= \nabla_{\boldsymbol{\theta}} \log \Pr(\boldsymbol{x}|\hat{\boldsymbol{\theta}}) && \text{Fisher score vector} \\ Z(\hat{\boldsymbol{\theta}}) &= \mathrm{E}_{\boldsymbol{x}}\left[\boldsymbol{s}(\boldsymbol{x},\hat{\boldsymbol{\theta}})\boldsymbol{s}(\boldsymbol{x},\hat{\boldsymbol{\theta}})^{\top}\middle|\hat{\boldsymbol{\theta}}\right] && \text{Fisher information matrix} \end{aligned}$$

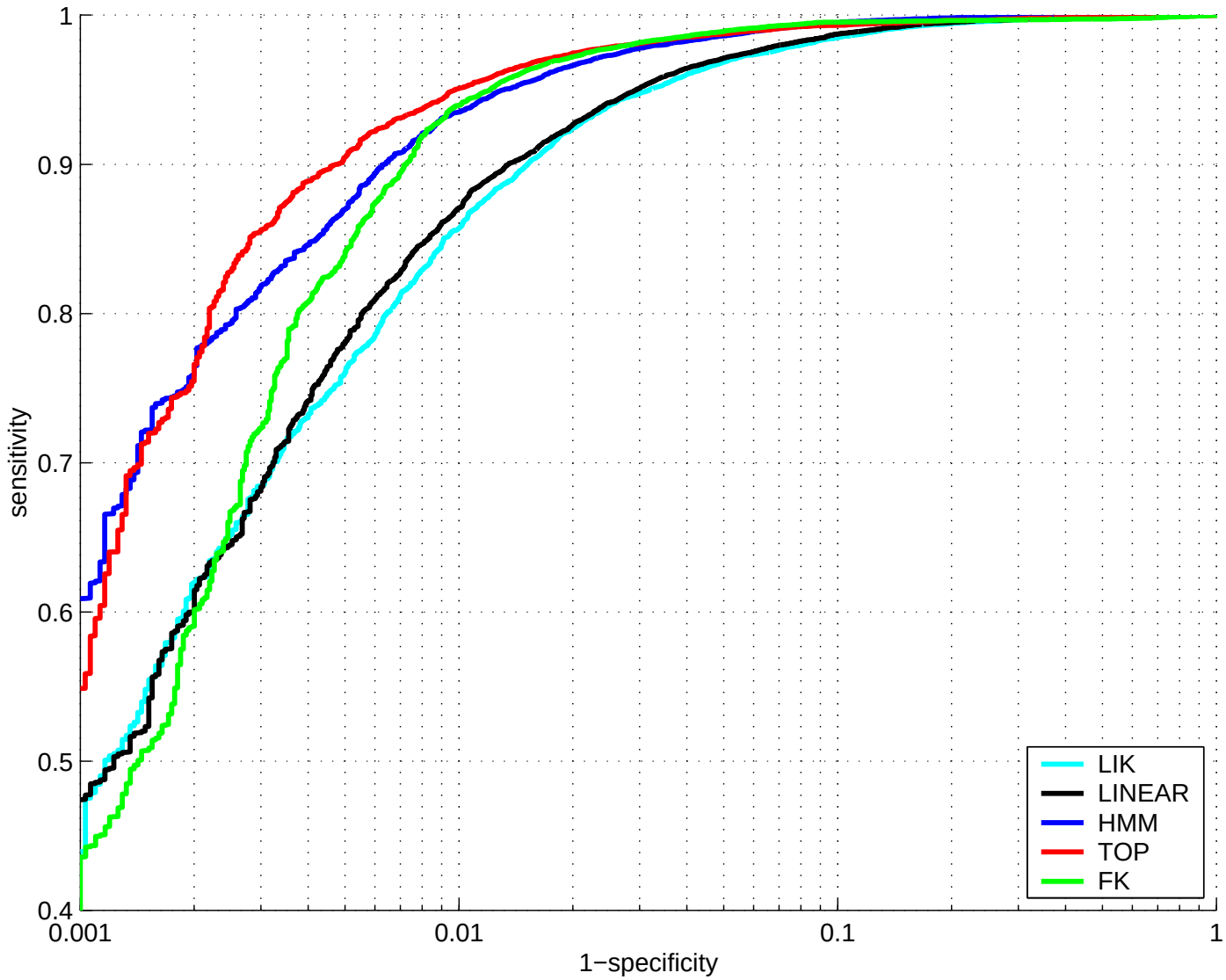
- TOP Kernel (Tsuda et. al, 2002)

$$\begin{aligned} \mathrm{k}_{\mathrm{TOP}}(\boldsymbol{x},\boldsymbol{x}') &= \boldsymbol{f}_{\hat{\boldsymbol{\theta}}}(\boldsymbol{x})^{\top} \boldsymbol{f}_{\hat{\boldsymbol{\theta}}}(\boldsymbol{x}') \\ \boldsymbol{f}_{\hat{\boldsymbol{\theta}}}(\boldsymbol{x}) &= (v(\boldsymbol{x},\hat{\boldsymbol{\theta}}), \nabla_{\theta}\, v(\boldsymbol{x},\hat{\boldsymbol{\theta}}))^{\top} \\ v(\boldsymbol{x},\hat{\boldsymbol{\theta}}) &= \log(\Pr(y=+1|\boldsymbol{x},\hat{\boldsymbol{\theta}})) - \log(\Pr(y=-1|\boldsymbol{x},\hat{\boldsymbol{\theta}})) \end{aligned}$$

Ergebnisse

System	Neither	Donor	Acceptor	Total
LI-SVM	2.0±0.3%	0.8±0.2%	0.9±0.3%	3.7%
FK-SVM	2.1±0.4%	1.6±0.5%	1.6±0.4%	5.3%
TOP-SVM	2.2±0.4%	1.5±0.4%	1.7±0.3%	5.4%
HMM	2.6±0.5%	1.0±0.4%	2.4±0.7%	6.0%
RBF-SVM	n.d.	n.d.	n.d.	>10%
NN-BRAIN	n.d.	2.6%	4.3%	n.d.
BRAIN	4.0%	5.0%	4.0%	13.0%
KBANN	4.6%	7.6%	8.5%	20.7%
BackProp	5.3%	5.7%	10.7%	21.7%
PEBLS	6.9%	8.2%	7.6%	22.7%
Perceptron	4.0%	16.3%	17.4%	37.7%
ID3	8.8%	10.6%	14.0%	33.4%
COBWEB	11.8%	15.0%	9.5%	36.3%
Near. Neigh.	31.1%	11.7%	9.1%	51.9%

Ergebnisse


1
0.9
0.8
0.7
0.6
0.5
0.4
sensitivity
0.001
0.01
0.1
1
1−specificity
LIK
LINEAR
HMM
TOP
FK