MULTI-CLASS PROTEIN CLASSIFICATION USING ADAPTIVE CODES

Jérôme Kirchhoff

Seminar maschinelles Lernen

Einleitung

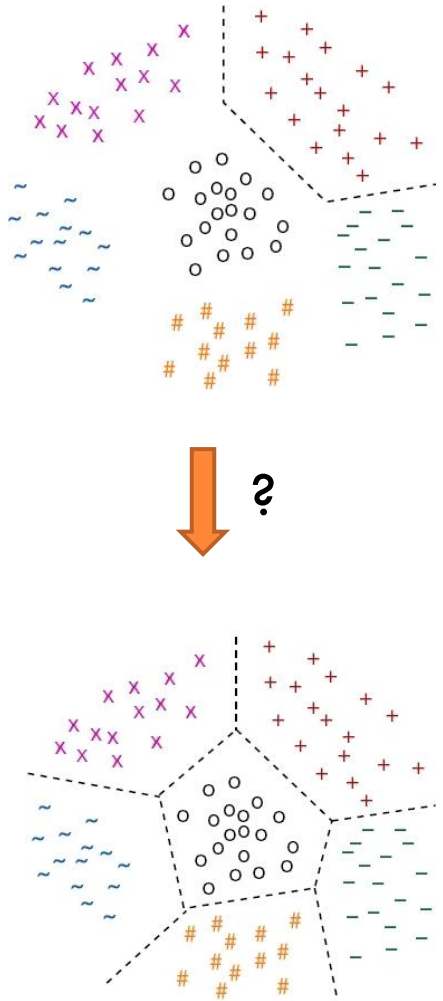
- Motivation
  - Ohne Proteine
    - kein Stoffwechsel
    - keine Zellteilung

      ⇒ kein Leben
  - Verstehen der Proteine → Verstehen des Lebewesens
  - Beschreibung aller Proteine nicht möglich
    - Charakterisierung über die Struktur
  - Proteinklassifizierung
    - Vorhersage der strukturellen Klassen
    - Strukturelle Kategorie lässt Funktion vorhersagen

Verfahren & Herangehensweisen

- Diskriminative Klassifizierer
  - Überragen bei Protein Klassifikation
  - Wie auf Multiklassenproblem anwenden?
- Ansatz
  - Problem auf binäre Klassifizierer reduzieren
  - Ausgabevektoren bearbeiten
- One-vs-all
  - Problem: Klassifizierer nicht vergleichbar
  - großer Vorhersagenswert ≠ beste Klasse

    → Sigmoid Anpassung bei vielen Daten
  - Hierarchische Beziehungen nicht berücksichtigt



Gliederung

Einleitung

Hintergrundwissen

Multiklassen Algorithmen

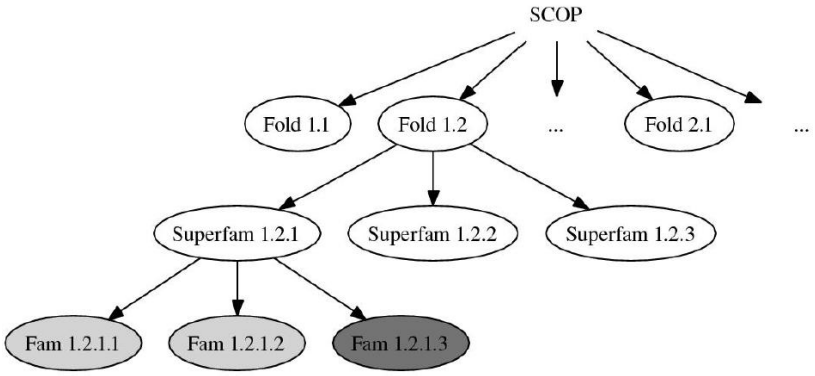
Experimentergebnisse

Zusammenfassung

Hintergrundwissen

Remote Homology Detection

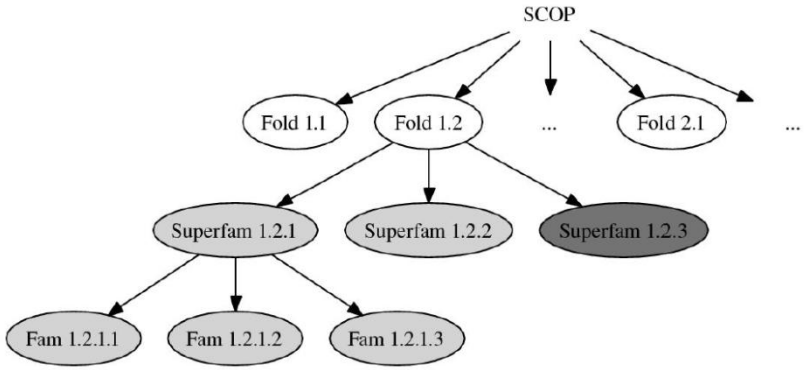
- Remote Homology Detection
  - Erkennen entfernter Verwandtschaften
  - Sequenzen aus Datenbank



- basierend auf SCOP

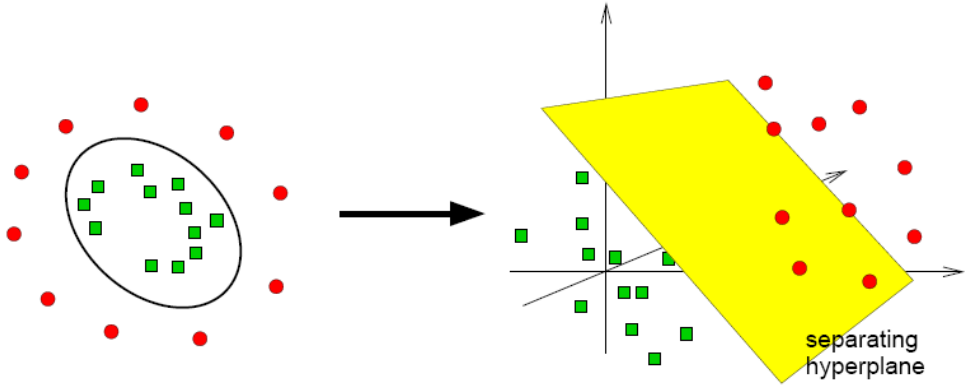
Fold Recognition

- Fold Recognition
  - Einteilung in Folds
  - Vergleichssequenzen in Strukturdatenbank
  - ohne evolutionäre Verwandschaft



Profile-Based Detectors

- Grundlegende binäre Klassifizierer
- Lernen in höherer Dimension durch Kernel
  - Kernel Funktion verhält sich wie Skalarprodukt in H



- Sequenzenprofil vergleichen
  - Positionsspezifische Verteilung
- K-mers

Positional Neighborhood

- Sequenz x hat Profil P(x)
- Für eine bestimmte Stelle j gilt, threshold σ
- Betrachten der k-mer $\beta = b_1 b_2 \cdots b_k$
  - Wahrscheinlichkeit der Position aus dem Profil
  - Addition dieser muss kleiner threshold sein
- Evtl. Variation der Aminosäuren wird bewertet
- Dies soll mögliche Mutationen im Gen berücksichtigen

Profile Feature Mapping

- Linear nicht trennbare Daten
- Transformation in den höherdimensionalen Raum
- Betrachtet werden die K-mers
  - Jede Position mit einer Aminosäure besetzt
  - Jede mögliche Kombination der Aminosäuren prüfen
  - Liegt sie in Neighborhood wird 1 im Vektor vermerkt
  - Somit ergibt sich ein Vektor mit feature Dimension $|\Sigma|^k$

Profile Kernel

- Profile Kernel lautet:

$$K_{(k,\sigma)}^{Profile}(P(x),P(y)) = \left\langle \Phi_{(k,\sigma)}^{Profile}(p(x)), \Phi_{(k,\sigma)}^{Profile}(p(y)) \right\rangle$$

  - Verhält sich wie Skalarprodukt in H
- Nicht nötig Feature-Raum zu kennen
- Skalarprodukt ermitteln ohne Φ anzuwenden
- Semi-Supervised Learning
  - Gekennzeichnete, sowie ungekennzeichnete Daten verwendet

PSI-BLAST

- Sequenzenvergleichsalgorithmus
- Iteratives Abgleichen mit Datenbanksequenzen
- Verwandte Sequenzen mitteln
- Profil erstellen
- Berücksichtigen entfernter Verwandtschaften
- Erneute Anfrage an Datenbank
- Genutzt für das Sequenzenprofil

Alignment:

```
  -  A  G  G  C  T  A  T  C  A  C  C  T  G
  T  A  G  -  C  T  A  C  C  A  -  -  -  G
  C  A  G  -  C  T  A  C  C  A  -  -  -  G
  C  A  G  -  C  T  A  T  C  A  C  -  G  G
  C  A  G  -  C  T  A  T  C  G  C  -  G  G
```

Profil:

```
A     1                 1        .8
C  .6          1           .4  1     .6 .2
G        1 .2                       .2        .4  1
T  .2                1     .6                 .2
-  .2       .8                          .4 .8 .4
```

Ausgabevektoren und Codes

- Hierarchie in Ausgaberepräsentation einbinden
- SCOP-Daten
  - k: Anzahl der Superfamilies
  - q: Anzahl der Folds
- Ein binärer Klassifizierer für jede Klasse

$$\vec{f}(x) = (f_1(x), \ldots, f_{k+q}(x))$$

$$C_j = (superfam_j, fold_j)$$

- Einführen von Code-Vektoren
- Ziel:
  - Gewichtungsvektor lernen, sodass $\mathbf{W} = (W_1, \ldots, W_{k+q})$
  - $\hat{y} = \arg\max_j \mathbf{W} \cdot (\vec{f}(x) * C_j)$ gilt

Zusammenfassung Hintergrundwissen

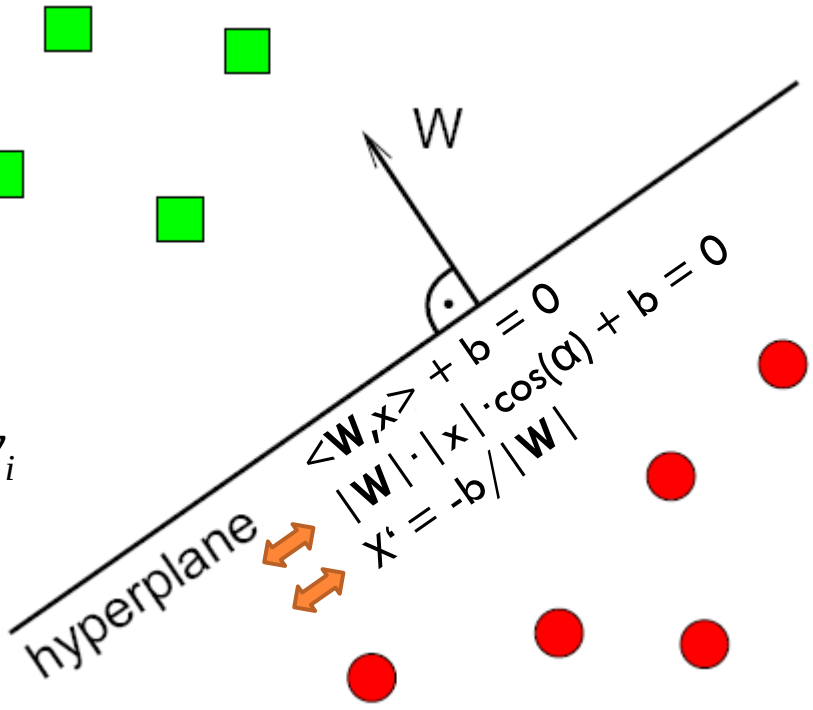
- Klassifizierer: Profile Based String Kernel SVM
- Profil generiert mit PSI-BLAST
- Hierarchische Struktur durch Codes eingebunden
- Gewichtung dieser soll Multiklassenvorhersage optimieren

Multiklassen Algorithmen

- Vorhersagensregel durch **W** anpassen
- Neugewichtung soll breiten Margin zwischen korrekten und falschen Codes herstellen
- „hard-margin“ Optimierungsproblem
- Margin: $\frac{1}{|\mathbf{w}|}$ ⇒ $\|\mathbf{w}\|^2$ minimieren
- Korrekte Trennung

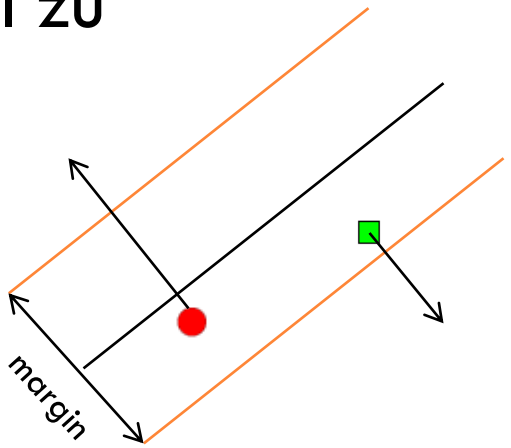
  durch Nebenbedingung

$$\mathbf{W} \cdot (\vec{f}(x_i) * C_{y_i} - \vec{f}(x_i) * C_j) \geq m, \ \forall\, j \neq y_i$$



Ranking Perceptron

- Eine Art linearer Klassifizierer
- Erhält Trainingsvektoren aus Kreuzvalidierung
- Produziert Gewichtungsvektor **W**
- Update-Regeln von Loss-Function abhängig
  - Zero-One loss: Fehler werden identisch gewertet
  - Balanced loss: Fehler inversproportional zu Klassengröße gewertet
- Ablauf: - **W** zu Beginn 0
  - Für angegebene Anzahl an Iterationen n:
    Was ist die momentane Klassifizierung?
    Ist diese im Margin, oder falsch klassifiziert?
    Wenn ja: update **W**



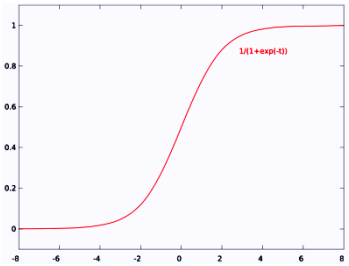
Update-Regeln

- Beziehungen zwischen Codes für Perceptron nutzen
- Update-Regel kann neu definiert werden
- Friend(y_i): Codes aus demselben Fold wie y_i
- Foes(y_i): alle die nicht in friend(y_i) enthalten sind
- Wenn schwächster Freund den stärksten Feind nicht durch einen Margin schlägt ⇨ Update
- Diese Regel heißt Friend/Foe Update
- Mean Friend/Foe nutzt jeweils mittleren Code

Vergleichen von Annäherungen

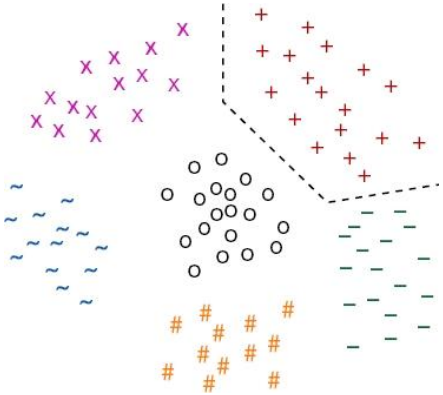
Platt's Sigmoid Methode

- SVM Ausgaben unbeschränkt
- Zudem auch nicht skaliert
- Konvertieren in klassenspezifische Wahrscheinlichkeiten
- Sigmoid anpassen
- Vergleichen der Ergebnisse nun möglich
- Schlechte Performanz bei kleinen Datenmengen



One-vs-all

- Soll gleichwertig zu allen Annäherungen sein
- Jedoch häufig leicht zerbrechlich
- Ein fehlerhafter Klassifizierer kann falsche Klassifikation hervorrufen
- Je mehr Klassen desto höher die Chance dafür



Zusammenfassung Multiklassen Algorithmen

- Lernen von **W** ist „hard margin“ Problem
- Durch Neugewichtung großen Margin zwischen korrekten und falschen Codes
- **W** durch Ranking Perceptron Algoithmus gelernt
- Modifizieren der Update-Regel durch Berücksichtigen der Beziehungen zwischen Codes
- Andere Annäherungen:
  - Sigmoid Anpassung
  - One-vs-all

Experimentergebnisse

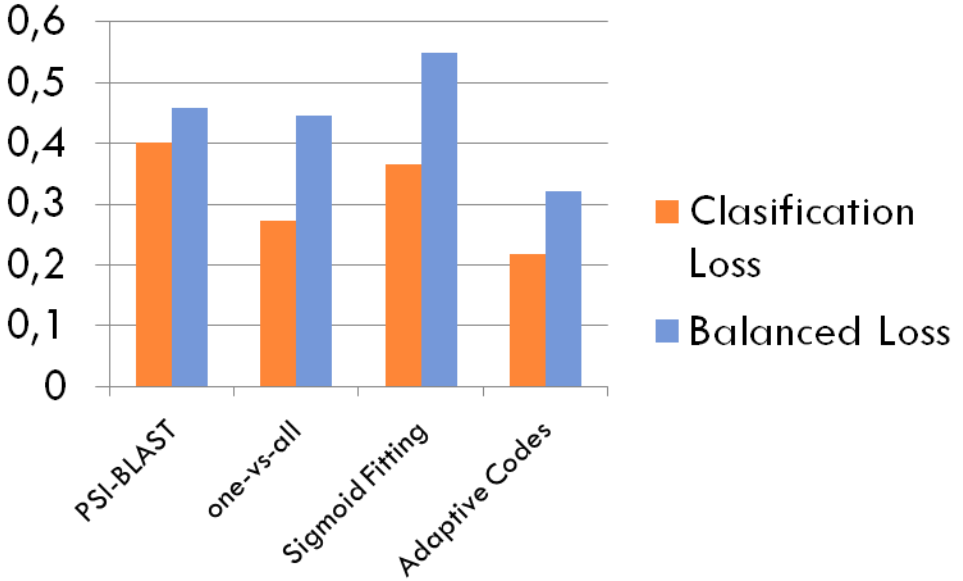
- Sequenzen aus SCOP 1.65 Proteindatenbank
- Preprocessing verwirft ähnliche Sequenzen
- Fold Recognition:
  - 26 Folds, 303 Superfamilies, 652 Families zum Trainieren
  - 614 „hold-out“ Sequenzen aus 46 Superfamilies zum Testen
- Remote Homology Detection
  - Superfamily: 74 Superfamilies, 54 Families zum Trainieren
    802 Sequenzen aus 110 Families zum Testen
  - Fold: 44 Folds, 424 Superfamilies , 809 Families zum Training
    381 Sequenzen aus 136 Families zum Testen
- Kreuzvalidierungswerte für Codegewichtung lernen
- Basisklassifizierer in zwei Stufen trainiert

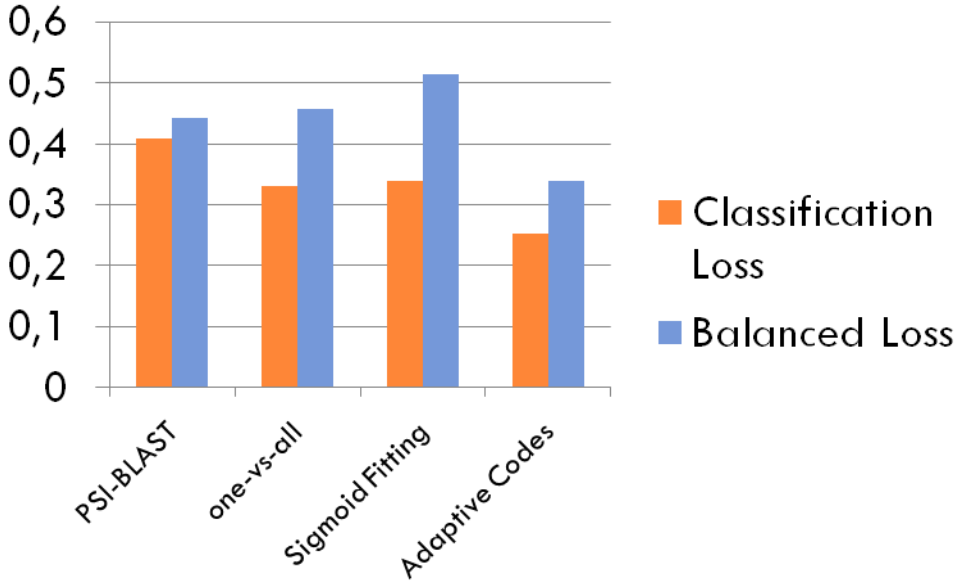
Methoden

- Ranking Perceptron testen mit:
  - Class-Based, Friend/Foe, Mean Class Update-Regeln
  - Trainiert mit 200 Iterationen
- Vergleichen unserer Methode mit:
  - One-vs-all
  - Sigmoid Anpassung
  - PSI-BLAST als Basismethode des Sequenzenvergleichs

Remote Homology Detection Auswertung

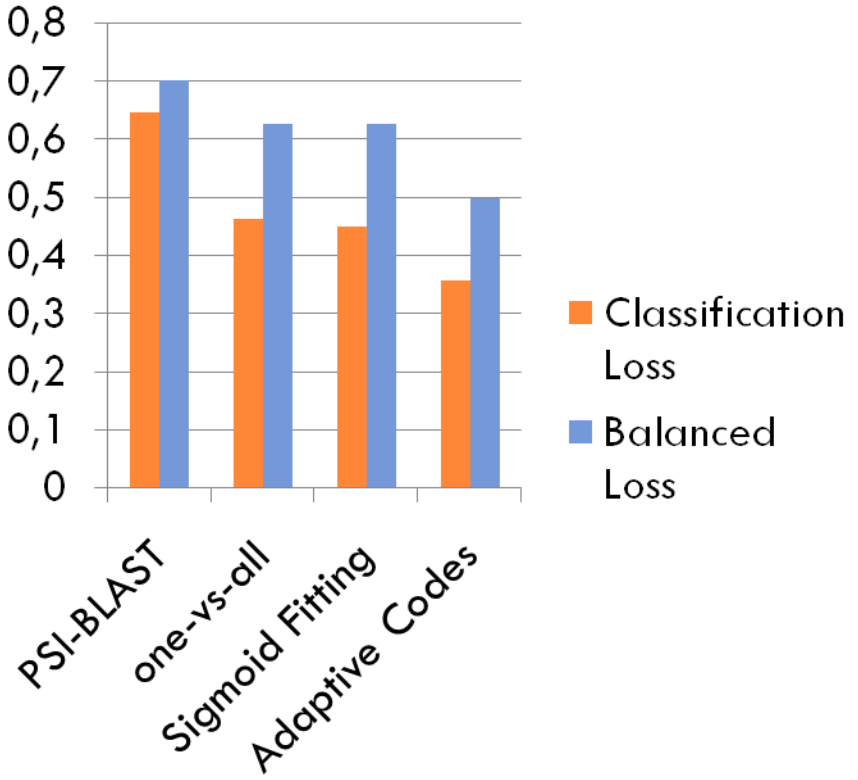
- Darstellen von Classification und Balanced Loss insgesamt und top5 (nicht dargestellt)
- Superfamily-Prediction:
  - Adaptive Codes besser als One-vs-all bzgl. beider Loss Funktionen
- Fold-Prediction:
  - Selber Trend
- Verbessert Performanz bei Nutzung mehrerer Codeelemente





Fold Recognition Auswertung

- Subklassen bieten evtl. weniger Informationen
- Proteinsequenzen aus verschieden Superfamilies haben keine erkennbare Gleichheit
- Adaptive Codes:
  - Besser als die anderen Verfahren
- Kein Verbesserungstrend mit zunehmender Anzahl von Codeelementen



Perceptron Update Rules Auswertung

- Beide Remote Homology Vorhersagen werden durch die Update-Regeln verbessert (bei zero-one und balanced loss)
  - Superfamily-Prediction Verbesserung, wenn Balanced-Loss trainiert und getestet
- Mehr Codeelemente für Fold Recognition wenig nützlich
  - Kann Performanz im Zero-One Loss nur verbessern, wenn mit Balanced Loss trainiert

Zusammenfassung Experimentergebnisse

- Remote Homology Detection
  - Superfamily- und Fold-prediction
    - Besser als verglichene Annäherungen, vor allem one-vs-all
    - Zudem stetige Verbesserung mit zusätzlichen Codeelementen
- Fold Recognition
  - Wieder besser als verglichene Annäherungen
  - Zusätzliche Codeelemente nicht hilfreich
    - Kein Verbesserungstrend mit längeren Codes
- Perceptron Update Rules
  - Remote Homology: Verbesserung möglich
  - Fold Recognition: Verbesserung minimal

Zusammenfassung

- Profile Based String Kernel SVMs klassifizieren
- Gewichtungsvektor W mit Ranking Perceptron lernen
  - Evtl. mit modifizierten Update-Regeln
- Ausgabevektor mit Gewichtung bearbeiten
- Somit Multiklassenvorhersage optimieren

- Ergebnis: Verfahren, welches die bisherigen schlägt
- Verbessert Annäherung für zwei ungelöste Probleme der Biologie

Vielen Dank für die Aufmerksamkeit

Quellen:

- Multi-class Protein Classification Using Adaptive Codes
  Iain Melvin, Eugene Ie, Jason Weston, William Stafford Noble, Christina Leslie
- Folien Lehrveranstaltung: Maschinelles Lernen
  Johannes Fürnkranz / Bent Schiele
- Folien Lehrveranstaltung: Bioinformatik I
  Martin Lercher, Ph.D.
- Klassikation mit Support Vector Machines
  Florian Markowetz, Max-Planck-Institut für Molekulare Genetik
- www.tobias-lib.ub.uni-tuebingen.de/volltexte/2005/1926
- www.wikipedia.org