PG 520
Supervised Clustering with SVM

Alexander Viefhues

Universität Dortmund 2007

Gliederung

Quelle

Thomas Finley
Thorsten Joachims
Supervised Clustering with Support Vector Machines
Department of Computer Science, Cornell University, Ithaca, NY 14853 USA

Motivation

Problem

gegeben: Itemsets fertig in Cluster unterteilt.

gesucht: Fähigkeit, alle folgenden Itemsets automatisch richtig in Cluster zu unterteilen.

Problem: Welche Objekte sind „ähnlich“ und kommen in ein Cluster?

Lösungsidee

Man benutz die SVM zum Trainieren des Cluster-Algorithmus.

Motivation

Problem

gegeben: Itemsets fertig in Cluster unterteilt.

gesucht: Fähigkeit, alle folgenden Itemsets automatisch richtig in Cluster zu unterteilen.

Problem: Welche Objekte sind „ähnlich“ und kommen in ein Cluster?

Lösungsidee

Man benutz die SVM zum Trainieren des Cluster-Algorithmus.

Supervised Clustering

Formal

gegeben: n Trainingssets : $(x_i, y_i)...(x_n, y_n) \in X \times Y$
X sind alle Items.
Y alle möglichen Partitionen von X.

gesucht: Clusterfunktion: $h: X \rightarrow Y$

Hilfsmittel

Verlustfunktion: $\Delta : Y \times Y \rightarrow \mathbb{R}$
vergleicht zwei Clusterings

Trainingserror: $\Delta(h(x), y)$
für ein Beispiel (x,y) und Clusterfunktion h

Supervised Clustering

Formal

gegeben: n Trainingssets : $(x_i, y_i)...(x_n, y_n) \in X \times Y$
X sind alle Items.
Y alle möglichen Partitionen von X.

gesucht: Clusterfunktion: $h: X \rightarrow Y$

Hilfsmittel

Verlustfunktion: $\Delta : Y \times Y \rightarrow \mathbb{R}$
vergleicht zwei Clusterings

Trainingserror: $\Delta(h(x), y)$
für ein Beispiel (x,y) und Clusterfunktion h

Paarweise Klassifikation

Funktionsweise

Paarweiser Vergleich der Items,
beschreiben der Paare durch einen Eigenschaftsvektor

Positive Beispiele sind die in einem Cluster.
Negative die, die nicht in einem Cluster sind.

Der Ausgabewert gibt an, ob die Items im selben Cluster sind oder nicht.

Paarweise Klassifikation

Probleme

- Wenn nicht eindeutig klar ist, ob ein Paar im gleichen Cluster ist oder nicht, kann der paarweise Klassifizierer nicht nach dem geforderten Leistungsmaß optimieren.
- Paare in einem Cluster oft sehr gering, deshalb Unterbewertung des Ergebnisses möglich
- Der Vorteil von Abhängigkeiten der einzelnen Paare kann nicht genutzt werden

Paarweise Klassifikation

Probleme

- Wenn nicht eindeutig klar ist, ob ein Paar im gleichen Cluster ist oder nicht, kann der paarweise Klassifizierer nicht nach dem geforderten Leistungsmaß optimieren.
- Paare in einem Cluster oft sehr gering,
  deshalb Unterbewertung des Ergebnisses möglich
- Der Vorteil von Abhängigkeiten der einzelnen Paare
  kann nicht genutzt werden

Paarweise Klassifikation

Probleme

- Wenn nicht eindeutig klar ist, ob ein Paar im gleichen Cluster ist oder nicht, kann der paarweise Klassifizierer nicht nach dem geforderten Leistungsmaß optimieren.
- Paare in einem Cluster oft sehr gering,
  deshalb Unterbewertung des Ergebnisses möglich
- Der Vorteil von Abhängigkeiten der einzelnen Paare
  kann nicht genutzt werden

Paarweise Klassifikation

Problemlösungsansätze

	Nur Heuristiken!
Cohen & Cardie:	lernen nur auf Paaren, die sich ähnlich sind.
Ng & Cardie:	basiert auf Zusammenhängen von Ausdrücken: $x_b + x_a$ werden pos. Paar, x_a ist der Ausdruck, der am nächsten vor x_b auftaucht und mit x_b in Verbindung steht. Alle Ausdrücke zwischen x_a und x_b werden mit x_b zu negativen Paaren

Das Modell

Grundsatz

An der Methode für das Supervides Clustering wird nichts geändert. Wir modifizieren nur das Ähnlichkeitsmaß, so dass der Algorithmus die erwünschten Ergebnisse liefert.

Ähnlichkeitsmaß

Das Ähnlichkeitsmaß $Sim_{\mathbf{w}}$ bildet Itempaare auf Werte ab
Positiv = Ähnlich; Negativ = Unähnlich
$\forall : (x_a, x_b); \quad x_a, x_b \in \mathrm{x}; \quad x_a \neq x_b \quad \exists : \varnothing(x_a, x_b) \equiv \varnothing_{a,b}$

Ähnlichkeitsmaß: $Sim_{\mathbf{w}}(x_a, x_b) = \mathrm{w}^T \varnothing_{a,b}$
$\varnothing$: Eigenschaftsvektor der Paare.
w: Parameter für das Ähnlichkeitsmaß.

Das Modell

Grundsatz

An der Methode für das Supervides Clustering wird nichts geändert. Wir modifizieren nur das Ähnlichkeitsmaß, so dass der Algorithmus die erwünschten Ergebnisse liefert.

Ähnlichkeitsmaß

Das Ähnlichkeitsmaß $Sim_{\mathbf{w}}$ bildet Itempaare auf Werte ab
Positiv = Ähnlich; Negativ = Unähnlich
$\forall : (x_a, x_b); \quad x_a, x_b \in \mathrm{x}; \quad x_a \neq x_b \quad \exists : \varnothing(x_a, x_b) \equiv \varnothing_{a,b}$

Ähnlichkeitsmaß: $Sim_{\mathbf{w}}(x_a, x_b) = \mathrm{w}^T \varnothing_{a,b}$
$\varnothing$: Eigenschaftsvektor der Paare.
w: Parameter für das Ähnlichkeitsmaß.

Das Modell

Methode

Für unsere Cluster Methode benutzen wir:
Corellation Clustering(Bansal 2002)

Für ein Itemset **x** wird das Clustering **y** gesucht,
so dass die Summe der Ähnlichkeitswerte maximal ist.

$\text{argmax}_{\mathbf{y}} \sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} Sim_{\mathbf{w}}(x_a, x_b)$

Hinweis

Durch dieses Vorgehen kann es vorkommen,
dass Paare, die eigentlich nicht in ein Cluster gehören,
zusammengefasst werden,
da der Netzeffekt einen Gesamtvorteil liefert.

Das Modell

Methode

Für unsere Cluster Methode benutzen wir:
Corellation Clustering(Bansal 2002)

Für ein Itemset **x** wird das Clustering **y** gesucht,
so dass die Summe der Ähnlichkeitswerte maximal ist.

$\text{argmax}_{\mathbf{y}} \sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} Sim_{\mathbf{w}}(x_a, x_b)$

Hinweis

Durch dieses Vorgehen kann es vorkommen,
dass Paare, die eigentlich nicht in ein Cluster gehören,
zusammengefasst werden,
da der Netzeffekt einen Gesamtvorteil liefert.

Das Modell

Umformung

$$\sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} Sim_{\mathbf{w}}(x_a, x_b)$$

$$= \sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} \mathbf{w}^T \varnothing_{a,b}$$

$$= \mathbf{w}^T \left(\sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} \varnothing_{a,b} \right)$$

Bemerkung

Die Zielfunktion ist ein linear Produkt
von $\mathbf{w}$ und einer Summe von $\varnothing$ Vektroren.

Auch wenn wir Corellation Clustering benutzen,
ist jede Zielfunktion, die ein linear Produkt von $\mathbf{w}$ ist, erlaubt.

Das Modell

Umformung

$$\sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} Sim_{\mathbf{w}}(x_a, x_b)$$

$$= \sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} \mathbf{w}^T \varnothing_{a,b}$$

$$= \mathbf{w}^T \left(\sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} \varnothing_{a,b} \right)$$

Bemerkung

Die Zielfunktion ist ein linear Produkt
von $\mathbf{w}$ und einer Summe von $\varnothing$ Vektroren.

Auch wenn wir Corellation Clustering benutzen,
ist jede Zielfunktion, die ein linear Produkt von $\mathbf{w}$ ist, erlaubt.

Das Modell

Umformung

$$\sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} Sim_{\mathbf{w}}(x_a, x_b)$$

$$= \sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} \mathbf{w}^T \varnothing_{a,b}$$

$$= \mathbf{w}^T \left(\sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} \varnothing_{a,b} \right)$$

Bemerkung

Die Zielfunktion ist ein linear Produkt
von $\mathbf{w}$ und einer Summe von $\varnothing$ Vektroren.

Auch wenn wir Corellation Clustering benutzen,
ist jede Zielfunktion, die ein linear Produkt von $\mathbf{w}$ ist, erlaubt.

Das Modell

Umformung

$$\sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} Sim_{\mathbf{w}}(x_a, x_b)$$

$$= \sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} \mathbf{w}^T \varnothing_{a,b}$$

$$= \mathbf{w}^T \left(\sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} \varnothing_{a,b} \right)$$

Bemerkung

Die Zielfunktion ist ein linear Produkt
von $\mathbf{w}$ und einer Summe von $\varnothing$ Vektroren.

Auch wenn wir Corellation Clustering benutzen,
ist jede Zielfunktion, die ein linear Produkt von $\mathbf{w}$ ist, erlaubt.

Der Algorithmus

SVM^{struct}

löst das quadratische Programm:

$\min_{\mathbf{w},\xi} \ \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i, s.t. \forall : \xi_i \geq 0$

$\forall i, \forall \mathbf{y} \in Y \backslash y_i : \mathbf{w}^T\Psi(x_i, y_i) \geq \mathbf{w}^T\Psi(x_i, y) + \Delta(y_i, y) - \xi_i$

Erklärung

$\Delta(y, \hat{y})$ ist der Wertverlust zwischen dem wirklichen Cluster y und dem Vorhergesagten $\hat{y}$

$\Delta(y, \hat{y}) = 0$ wenn die Cluster gleich sind, Δ steigt, je unterschiedlicher die Cluster werden.

Erklärung

$\Psi(x_i, y) = \frac{1}{|x_i|^2}\sum_{y\in\mathbf{y}}\sum_{x_a,x_b\in y}\varnothing(x_a, x_b)$

Der Algorithmus

SVM^{struct}

löst das quadratische Programm:

$\min_{\mathbf{w},\xi} \ \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i, s.t. \forall : \xi_i \geq 0$

$\forall i, \forall \mathrm{y} \in Y \backslash y_i : \mathbf{w}^T\Psi(x_i, y_i) \geq \mathbf{w}^T\Psi(x_i, y) + \Delta(y_i, y) - \xi_i$

Erklärung

$\Delta(y, \hat{y})$ ist der Wertverlust zwischen dem wirklichen Cluster y und dem Vorhergesagten $\hat{y}$

$\Delta(y, \hat{y}) = 0$ wenn die Cluster gleich sind, Δ steigt, je unterschiedlicher die Cluster werden.

Erklärung

$\Psi(x_i, y) = \frac{1}{|x_i|^2}\sum_{y\in\mathbf{y}}\sum_{x_a,x_b\in y}\varnothing(x_a, x_b)$

Der Algorithmus

SVM^{struct}

löst das quadratische Programm:

$\min_{\mathbf{w},\xi} \ \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i, s.t. \forall : \xi_i \geq 0$

$\forall i, \forall \mathrm{y} \in Y \backslash y_i : \mathbf{w}^T\Psi(x_i, y_i) \geq \mathbf{w}^T\Psi(x_i, y) + \Delta(y_i, y) - \xi_i$

Erklärung

$\Delta(y, \hat{y})$ ist der Wertverlust zwischen dem wirklichen Cluster y und dem Vorhergesagten $\hat{y}$

$\Delta(y, \hat{y}) = 0$ wenn die Cluster gleich sind, Δ steigt, je unterschiedlicher die Cluster werden.

Erklärung

$\Psi(x_i, y) = \frac{1}{|x_i|^2}\sum_{y\in\mathbf{y}}\sum_{x_a,x_b\in y}\varnothing(x_a, x_b)$

Der Algorithmus

Bemerkung

Das QP führt für jedes falsche Clustering eine Nebenbedingung ein.

Jedoch wächst die Anzahl der falschen Clusterings mehr als exponentiell mit der Anzahl der Items.

Lösungsansatz

Die SVM beginnt mit keiner Nebenbedingung
und findet dann iterativ immer die Nebenbedingung, die am meisten verletzt wird.

Der Algorithmus

Bemerkung

Das QP führt für jedes falsche Clustering eine Nebenbedingung ein.

Jedoch wächst die Anzahl der falschen Clusterings mehr als exponentiell mit der Anzahl der Items.

Lösungsansatz

Die SVM beginnt mit keiner Nebenbedingung
und findet dann iterativ immer die Nebenbedingung, die am meisten verletzt wird.

Der Algorithmus für $SVM_1^{\Delta m}$

1: Imput: $(x_1, y_1), ..., (x_n, y_n), C, \epsilon$
2: $S_i \leftarrow \emptyset$ für alle $i = 1, ..., n$
3: **repeat**
4: **for** $i = 1, ..., n$ **do**
5: $H(y) \equiv \Delta(y_i, y) + \mathbf{w}^T \Psi(x_i, y) - \mathbf{w}^T \Psi(x_i, y_i)$
6: berechne: $\hat{y} = \text{argmax}_{y \in Y} H(y)$
7: berechne: $\xi_i = \max\{0, \max_{y \in S_i} H(y)\}$
8: **if** $H(\hat{y}) > \xi_i + \epsilon$ **then**
9: $S_i \leftarrow S_i \cup \{\hat{y}\}$
10: $\mathbf{w} \leftarrow$ Optimiere über $S = \bigcup_i S_I$
11: **end if**
12: **end for**
13: **until** kein S_i hat sich während des Duchlaufs geändert

Der Algorithmus

Der Algorithmus

Imput: $(x_1, y_1), \ldots, (x_n, y_n), C, \epsilon$
$S_i \leftarrow \emptyset$ für alle $i = 1, \ldots, n$
repeat
  for $i = 1, \ldots, n$ **do**
    $H(y) \equiv \Delta(y_i, y) + \mathbf{w}^T \Psi(x_i, y) - \mathbf{w}^T \Psi(x_i, y_i)$
    berechne: $\hat{y} = \text{argmax}_{y \in Y} H(y)$
    berechne: $\xi_i = \max\{0, \max_{y \in S_i} H(y)\}$
    if $H(\hat{y}) > \xi_i + \epsilon$ **then**
      $S_i \leftarrow S_i \cup \{\hat{y}\}$
      $\mathbf{w} \leftarrow$ Optimiere über $S = \bigcup_i S_I$
    end if
  end for
until kein S_i hat sich während des Duchlaufs geändert

Erläuterung

$\text{argmax}_{y \in Y} H(y)$ liefert die am meisten verletze Nebenbedingung.

Wenn die Nebenbedingung mehr als ϵ verletzt wird, wird sie übernommen

und es wird neu optimiert.

Wiederholung solange, bis keine Nebenbedingung mehr hinzukommt.

Der Algorithmus

Der Algorithmus

Imput: $(x_1, y_1), \ldots, (x_n, y_n), C, \epsilon$
$S_i \leftarrow \emptyset$ für alle $i = 1, \ldots, n$
repeat
  for $i = 1, \ldots, n$ **do**
    $H(y) \equiv \Delta(y_i, y) + \mathbf{w}^T \Psi(x_i, y) - \mathbf{w}^T \Psi(x_i, y_i)$
    berechne: $\hat{y} = \text{argmax}_{y \in Y} H(y)$
    berechne: $\xi_i = \max\{0, \max_{y \in S_i} H(y)\}$
    if $H(\hat{y}) > \xi_i + \epsilon$ **then**
      $S_i \leftarrow S_i \cup \{\hat{y}\}$
      $\mathbf{w} \leftarrow$ Optimiere über $S = \bigcup_i S_I$
    end if
  end for
until kein S_i hat sich während des Duchlaufs geändert

Erläuterung

$\text{argmax}_{y \in Y} H(y)$ liefert die am meisten verletze Nebenbedingung.

Wenn die Nebenbedingung mehr als ϵ verletzt wird, wird sie übernommen

und es wird neu optimiert.

Wiederholung solange, bis keine Nebenbedingung mehr hinzukommt.

Der Algorithmus

Der Algorithmus

Imput: $(x_1, y_1), \ldots, (x_n, y_n), C, \epsilon$
$S_i \leftarrow \emptyset$ für alle $i = 1, \ldots, n$
repeat
  for $i = 1, \ldots, n$ **do**
    $H(y) \equiv \Delta(y_i, y) + \mathbf{w}^T \Psi(x_i, y) - \mathbf{w}^T \Psi(x_i, y_i)$
    berechne: $\hat{y} = \text{argmax}_{y \in Y} H(y)$
    berechne: $\xi_i = \max\{0, \max_{y \in S_i} H(y)\}$
    if $H(\hat{y}) > \xi_i + \epsilon$ **then**
      $S_i \leftarrow S_i \cup \{\hat{y}\}$
      $\mathbf{w} \leftarrow$ Optimiere über $S = \bigcup_i S_I$
    end if
  end for
until kein S_i hat sich während des Duchlaufs geändert

Erläuterung

$\text{argmax}_{y \in Y} H(y)$ liefert die am meisten verletze Nebenbedingung.

Wenn die Nebenbedingung mehr als ϵ verletzt wird, wird sie übernommen

und es wird neu optimiert.

Wiederholung solange, bis keine Nebenbedingung mehr hinzukommt.

Der Algorithmus

Der Algorithmus

Imput: $(x_1, y_1), \ldots, (x_n, y_n), C, \epsilon$
$S_i \leftarrow \emptyset$ für alle $i = 1, \ldots, n$
repeat
  for $i = 1, \ldots, n$ **do**
    $H(y) \equiv \Delta(y_i, y) + \mathbf{w}^T \Psi(x_i, y) - \mathbf{w}^T \Psi(x_i, y_i)$
    berechne: $\hat{y} = \text{argmax}_{y \in Y} H(y)$
    berechne: $\xi_i = \max\{0, \max_{y \in S_i} H(y)\}$
    if $H(\hat{y}) > \xi_i + \epsilon$ **then**
      $S_i \leftarrow S_i \cup \{\hat{y}\}$
      $\mathbf{w} \leftarrow$ Optimiere über $S = \bigcup_i S_i$
    end if
  end for
until kein S_i hat sich während des Duchlaufs geändert

Erläuterung

$\text{argmax}_{y \in Y} H(y)$ liefert die am meisten verletze Nebenbedingung.

Wenn die Nebenbedingung mehr als ϵ verletzt wird, wird sie übernommen

und es wird neu optimiert.

Wiederholung solange, bis keine Nebenbedingung mehr hinzukommt.

Der Algorithmus

Der Algorithmus

Imput: $(x_1, y_1), \ldots, (x_n, y_n), C, \epsilon$
$S_i \leftarrow \emptyset$ für alle $i = 1, \ldots, n$
repeat
  for $i = 1, \ldots, n$ **do**
    $H(y) \equiv \Delta(y_i, y) + \mathbf{w}^T \Psi(x_i, y) - \mathbf{w}^T \Psi(x_i, y_i)$
    berechne: $\hat{y} = \text{argmax}_{y \in Y} H(y)$
    berechne: $\xi_i = \max\{0, \max_{y \in S_i} H(y)\}$
    if $H(\hat{y}) > \xi_i + \epsilon$ **then**
      $S_i \leftarrow S_i \cup \{\hat{y}\}$
      $\mathbf{w} \leftarrow$ Optimiere über $S = \bigcup_i S_I$
    end if
  end for
until kein S_i hat sich während des Duchlaufs geändert

Erläuterung

$\text{argmax}_{y \in Y} H(y)$ liefert die am meisten verletze Nebenbedingung.

Wenn die Nebenbedingung mehr als ϵ verletzt wird, wird sie übernommen

und es wird neu optimiert.

Wiederholung solange, bis keine Nebenbedingung mehr hinzukommt.

Der Algorithmus

Theorem 1

Wenn $\overline{\Delta} = max_{(x_i,y_i)\in S}(max_y \Delta(y_i, y))$

und $\overline{R} = max_{(x_i,y_i)\in S}(max_y ||\Psi(x_i, y_i) - \Psi(x_i, y)||)$

dann konvergiert die SVM^{struct} nach $max\{\frac{2n\overline{\Delta}}{\epsilon}, \frac{8Cn\overline{\Delta}\overline{R}^2}{\epsilon^2}\}$

Theorem 2

Der Algorithmus liefert ein approximiertes Ergebnis, das kleiner oder gleich dem QP ist.

Alle Nebenbedingungen sind im Rahmen von ϵ erfüllt.

Der Algorithmus

Theorem 1

Wenn $\overline{\Delta} = max_{(x_i,y_i)\in S}(max_y \Delta(y_i, y))$

und $\overline{R} = max_{(x_i,y_i)\in S}(max_y ||\Psi(x_i, y_i) - \Psi(x_i, y)||)$

dann konvergiert die SVM^{struct} nach $max\{\frac{2n\overline{\Delta}}{\epsilon}, \frac{8Cn\overline{\Delta}\overline{R}^2}{\epsilon^2}\}$

Theorem 2

Der Algorithmus liefert ein approximiertes Ergebnis, das kleiner oder gleich dem QP ist.

Alle Nebenbedingungen sind im Rahmen von ϵ erfüllt.

Approximation

Das Problem

Wie schwer ist es, die meist verletzte Nebenbedingung zu finden?
$H(y) \equiv \Delta(y_i, y) + \mathbf{w}^T \Psi(x_i, y) - \mathbf{w}^T \Psi(x_i, y_i)$

Der letzte Teil bleibt gleich, und kann vernachlässigt werden, so dass sich das Maximum nicht ändert

Die Kostenfunktion: Verlust zwischen korrekten Clustering y_i und dem Vorhergesagten y

plus die Zielfunktion des Correkation Clustering.

Das Finden des y für die Zielfunktion ist NP-vollständig(Bansal,2002) den Verlust zu Addieren bringt keinen Vorteil.

Lösungsansatz

Approximation der Clusterzielfunktion und von $\text{argmax}_y H(y)$

Approximation

Das Problem

Wie schwer ist es, die meist verletzte Nebenbedingung zu finden?
$H(y) \equiv \Delta(y_i, y) + \mathbf{w}^T \Psi(x_i, y) - \mathbf{w}^T \Psi(x_i, y_i)$

Der letzte Teil bleibt gleich, und kann vernachlässigt werden, so dass sich das Maximum nicht ändert

Die Kostenfunktion: Verlust zwischen korrekten Clustering y_i und dem Vorhergesagten y

plus die Zielfunktion des Correkation Clustering.

Das Finden des y für die Zielfunktion ist NP-vollständig(Bansal,2002) den Verlust zu Addieren bringt keinen Vorteil.

Lösungsansatz

Approximation der Clusterzielfunktion und von $\text{argmax}_y H(y)$

Approximation

Greedy Approximation C_G

Start: jedes Item in eigenem Cluster

Finde zwei Cluster und vereinige sie, so dass $H(y)$ maximal erhöht wird.

Ende: wenn kein Zusammenfügen eine Erhöhung bringt.

Korollar

C_G Zielwert nicht größer als der von QP
wenn $\text{argmax}_y H(y) = \hat{y}$, findet C_G $\text{argmax}_y H(y) = y^*$ dann ist $H(\hat{y}) \geq H(y^*)$

Approximation

Greedy Approximation C_G

Start: jedes Item in eigenem Cluster

Finde zwei Cluster und vereinige sie, so dass $H(y)$ maximal erhöht wird.

Ende: wenn kein Zusammenfügen eine Erhöhung bringt.

Korollar

C_G Zielwert nicht größer als der von QP
wenn $\text{argmax}_y H(y) = \hat{y}$, findet C_G $\text{argmax}_y H(y) = y^*$ dann ist $H(\hat{y}) \geq H(y^*)$

Relaxation Approximation, C_R

Jedes Paar x_a, x_b hat eine Variable $e_{a,b}$ die anzeigt, zu welchem Maß die Items in das gleiche Cluster gehören.

$e_{a,b} \in [0, 1]$

$e_{a,b} = e_{b,a}$

$e_{a,b} + e_{b,c} \geq e_{a,c}$

LP: $\max_e \sum_{e_{a,b} \in \mathbf{e}} (e_{a,b}) \cdot (\mathbf{w}^T \varnothing_{a,b})$

$\Psi(x, e) = \frac{1}{x} \sum_{e_{a,b} \in e} (e_{a,b}) \cdot \varnothing_{a,b}$

Bemerkung: ist das Gleiche wie

$\Psi(x_i, y) = \frac{1}{|x_i|^2} \sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} \varnothing(x_a, x_b)$,

wenn $e_{a,b}$ ganzzahlig ist.

Korollar

C_R Zielwert nicht kleiner als der von QP

$H(\hat{y}) \leq H(e)$

Relaxation Approximation, C_R

Jedes Paar x_a, x_b hat eine Variable $e_{a,b}$ die anzeigt, zu welchem Maß die Items in das gleiche Cluster gehören.

$e_{a,b} \in [0, 1]$

$e_{a,b} = e_{b,a}$

$e_{a,b} + e_{b,c} \geq e_{a,c}$

LP: $\max_e \sum_{e_{a,b} \in \mathbf{e}} (e_{a,b}) \cdot (\mathbf{w}^T \varnothing_{a,b})$

$\Psi(x, e) = \frac{1}{x} \sum_{e_{a,b} \in e} (e_{a,b}) \cdot \varnothing_{a,b}$

Bemerkung: ist das Gleiche wie

$\Psi(x_i, y) = \frac{1}{|x_i|^2} \sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} \varnothing(x_a, x_b)$,

wenn $e_{a,b}$ ganzzahlig ist.

Korollar

C_R Zielwert nicht kleiner als der von QP

$H(\hat{y}) \leq H(e)$

Approximation

Bemerkung

Für die Experimente wurde C_R^* benutzt.

Start: jedes Item in eigenem Cluster

Iteriere über alle $x_a \in x$, wenn x_a alleine im Cluster wähle es aus

Für alle $x_b \in x$ die auch alleine in einem Cluster sind:

Cluster Vereinigen, wenn $e_{a,b} > 0,7$

Die Verlustfunktionen

Paarweise Δ_P

$\Delta(y, \overline{y}) = 100\frac{W}{T}$
T = Anzahl von Paaren im Itemset
W = Anzahl von Paaren, bei denen y und $\overline{y}$ nicht übereinstimmen.
Bei Relaxation Approximation: $e_{a,b} \cdot (\mathbf{w}^T \varnothing_{a,b} + \frac{100}{T})$

MITRE VerlustΔ_M

$\Delta_M(y, \overline{y}) = 100\frac{2(1-R)(1-P)}{(1-R)+(1-P)}$
R = MITRE recall
P = MITRE precision

Die Verlustfunktionen

Paarweise Δ_P

$\Delta(y, \overline{y}) = 100\frac{W}{T}$
T = Anzahl von Paaren im Itemset
W = Anzahl von Paaren, bei denen y und $\overline{y}$ nicht übereinstimmen.
Bei Relaxation Approximation: $e_{a,b} \cdot (\mathbf{w}^T \varnothing_{a,b} + \frac{100}{T})$

MITRE VerlustΔ_M

$\Delta_M(y, \overline{y}) = 100\frac{2(1-R)(1-P)}{(1-R)+(1-P)}$
R = MITRE recall
P = MITRE precision

Testmaterial

Testmaterial 1:MUC-6 noun-phrase

60 Dokumente(30 Training, 30 Test)

Jedes Dokument im Schnitt 101 Cluster

Jedes Cluster im Schnitt 1,48 Wörter,viele Einzelwort-Cluster

Eigenschaftsvektor der Paare hat 53 Dimensionen:
gleicher Artikel? Satzabstand der Worte!...

Testmaterial 2: Google News

30 Tage(15 Training, 15 Test) die Top 10 Themen der „Welt“ Kathegorie ausgewählt

von jeder Kathegorie meistens 15 Artikel ausgewählt

jeder Artikel hat 30 TFIDF Vektoren für: Unigramme, Bigramme....

Der Eigenschaftsvektor der Paare besteht aus den 30 „cosine similarities“ der Vektoren der Items

plus einem Element, das immer 1 ist.

Testmaterial

Testmaterial 1:MUC-6 noun-phrase

60 Dokumente(30 Training, 30 Test)

Jedes Dokument im Schnitt 101 Cluster

Jedes Cluster im Schnitt 1,48 Wörter,viele Einzelwort-Cluster

Eigenschaftsvektor der Paare hat 53 Dimensionen:
gleicher Artikel? Satzabstand der Worte!...

Testmaterial 2: Google News

30 Tage(15 Training, 15 Test) die Top 10 Themen der „Welt“ Kathegorie ausgewählt

von jeder Kathegorie meistens 15 Artikel ausgewählt

jeder Artikel hat 30 TFIDF Vektoren für: Unigramme, Bigramme....

Der Eigenschaftsvektor der Paare besteht aus den 30 „cosine similarities“ der Vektoren der Items

plus einem Element, das immer 1 ist.

Testumgebung

Testumgebung

Mit den Daten wurden verschiedene Modelle getestet.

ein Modell:

Ähnlichkeitsgewicht w

Regulierungsparameter C, ausgesucht aus verschiedenen Werten basierent auf Kreutzvalidierung(bei TM1 k=10, bei TM2 k =5)

Als Vergleich PCC(pairwise classifikation clustering) durch SVM_{light} realisiert

Testergebnisse bei MUC-6

	C_G	PCC	Default
Test mit C_G, Δ_M	41,3	51,6	51,0
Test mit C_G, Δ_P	2,89	3,15	3,59

Spalte C_G : - mit SVM-Cluster und Greedy Algorithmus gelernt
Spalte PCC: - mit PCC gelernt
Spalte Default: - in der ersten Zeile alle Items in einem Cluster
- in der zweiten jedes Item in eigenem Cluster

Zeile 1: getestet und optimiert mit MITRE Verlustfunktion
Zeile 2: getestet und optimiert mit paarweiser Verlustfunktion
Beide Tests benutzen den Greedy Algorithmus auf der Testmenge

Die SVM ist deutlich besser als der Rest.
Bei Δ_M kann man das dadurch erklären, dass die SVM für eine Verlustfunktion optimiert werden kann, PCC nicht.

Bei Δ_P lässt sich so nicht argumentieren, da PCC quasi gleich optimiert, trotzdem ist die SVM besser.

Testergebnisse bei MUC-6

	C_G	PCC	Default
Test mit C_G, Δ_M	41,3	51,6	51,0
Test mit C_G, Δ_P	2,89	3,15	3,59

Spalte C_G : - mit SVM-Cluster und Greedy Algorithmus gelernt
Spalte PCC: - mit PCC gelernt
Spalte Default: - in der ersten Zeile alle Items in einem Cluster
- in der zweiten jedes Item in eigenem Cluster

Zeile 1: getestet und optimiert mit MITRE Verlustfunktion
Zeile 2: getestet und optimiert mit paarweiser Verlustfunktion
Beide Tests benutzen den Greedy Algorithmus auf der Testmenge

Die SVM ist deutlich besser als der Rest.
Bei Δ_M kann man das dadurch erklären, dass die SVM für eine Verlustfunktion optimiert werden kann, PCC nicht.

Bei Δ_P lässt sich so nicht argumentieren, da PCC quasi gleich optimiert, trotzdem ist die SVM besser.

Testergebnisse bei MUC-6

	C_G	PCC	Default
Test mit C_G, Δ_M	41,3	51,6	51,0
Test mit C_G, Δ_P	2,89	3,15	3,59

Spalte C_G : - mit SVM-Cluster und Greedy Algorithmus gelernt
Spalte PCC: - mit PCC gelernt
Spalte Default: - in der ersten Zeile alle Items in einem Cluster
- in der zweiten jedes Item in eigenem Cluster

Zeile 1: getestet und optimiert mit MITRE Verlustfunktion
Zeile 2: getestet und optimiert mit paarweiser Verlustfunktion
Beide Tests benutzen den Greedy Algorithmus auf der Testmenge

Die SVM ist deutlich besser als der Rest.
Bei Δ_M kann man das dadurch erklären, dass die SVM für eine Verlustfunktion optimiert werden kann, PCC nicht.

Bei Δ_P lässt sich so nicht argumentieren, da PCC quasi gleich optimiert, trotzdem ist die SVM besser.

Testergebnisse bei Google News

	C_G	C_R	PCC	Default
Test mit C_G, Δ_P	2,36	2,43	2,45	9,45
Test mit C_R, Δ_P	2,04	2,08	1,96	9,45

Keine Mehtode ist deutlich besser als die andere.

Schlussfolgerung: Die SVM scheint besser zu sein, wenn die Daten transitive Abhängigkeiten haben, wenn nicht, sind beide Methoden annähernd gleich.

Optimierung für Verlustfunktionen

	Opt. to Δ_M	Opt. to Δ_P
Performance bei Δ_M	41,3	42,8
Performance bei Δ_P	4,06	2,89

Kaum Unterschiede bei Δ_M Performance.
Große Unterschiede bei Δ_P Performance.
Die Werte des für Δ_M optimierten Modells sind nicht besser, als die der Defaultwerte.
Schlussfolgerung: Die Wahl der Verlustfunktion kann nachhaltig die Accurracy des Modells beeinflussen.

Verlust bei $\text{argmax}_y H(y)$

Muss die Verlustfunktion in $\text{argmax}_y H(y)$ integriert sein, oder dürfen wir sie hier vernachlässigen?

	mit Verlusfunkt.	ohne Verlustfunkt.
MUC-6, Δ_M	41,3	41,1
MUC-6, Δ_P	2,89	2,81
Google, train C_G, test C_G	2,36	2,42
Google, train C_R, test C_R^*	2,08	2,16

Es gibt keine signifikante Änderung in den Zeilen, bei keinem Modell.
Schlussfolgerung: Verlust in $\text{argmax}_y H(y)$ nicht zwingend erforderlich.

Greedy vs. Relaxation

Welche Approximation ist besser?

	Train C_G	Train C_R
Test C_G	2,36	2,43
Test C_R	2,04	2,08

Kaum Unterschiede zu erkennen.
Schlussfolgerung: Keine der Approximationen ist zu bevorzugen.

Effizienz der SVM^{cluster}

Auf dem MUC-6 Material werden über 1000 Nebenbedingungen integriert und es wird 50 mal neu optimiert.
Mit C_G wurde nur 1% der Zeit benötigt.
Bei allen Experiementen hat keine SVM^{cluster} länger als 3-4 Stunden gebaucht,
wobei der Durchschnitt eher unter einer Stunde lag.
Als Vergleich der paarweise Klassifizierer:
- eine knappe halbe Million Paare im Trainingsset
- eine halbe Woche
- eine halbe Million Nebenbedingungen

Zusammenfassung

SVM^{cluster} vs. PCC

SVM je nach Material gleich oder besser als PCC.

Optimierung für Verlustfunktionen

Wahl der Verlustfunktion entscheidend für Accurracy, abhängig vom Modell.

Verlust bei argmax$_y H(y)$

Verlust in argmax$_y H(y)$ nicht zwingend erforderlich.

Greedy vs. Relaxation

Keine Approximation eindeutig besser.

Effizienz der SVM^{cluster}

SVM^{cluster} mit Approximation für die Geschwindigkeit die beste Wahl.

Zusammenfassung

SVM^{cluster} vs. PCC

SVM je nach Material gleich oder besser als PCC.

Optimierung für Verlustfunktionen

Wahl der Verlustfunktion entscheidend für Accurracy, abhängig vom Modell.

Verlust bei $\text{argmax}_y H(y)$

Verlust in $\text{argmax}_y H(y)$ nicht zwingend erforderlich.

Greedy vs. Relaxation

Keine Approximation eindeutig besser.

Effizienz der SVM^{cluster}

SVM^{cluster} mit Approximation für die Geschwindigkeit die beste Wahl.

Zusammenfassung

SVM^{cluster} vs. PCC

SVM je nach Material gleich oder besser als PCC.

Optimierung für Verlustfunktionen

Wahl der Verlustfunktion entscheidend für Accurracy, abhängig vom Modell.

Verlust bei argmax$_y H(y)$

Verlust in argmax$_y H(y)$ nicht zwingend erforderlich.

Greedy vs. Relaxation

Keine Approximation eindeutig besser.

Effizienz der SVM^{cluster}

SVM^{cluster} mit Approximation für die Geschwindigkeit die beste Wahl.

Zusammenfassung

SVM^{cluster} vs. PCC

SVM je nach Material gleich oder besser als PCC.

Optimierung für Verlustfunktionen

Wahl der Verlustfunktion entscheidend für Accurracy, abhängig vom Modell.

Verlust bei argmax$_y H(y)$

Verlust in argmax$_y H(y)$ nicht zwingend erforderlich.

Greedy vs. Relaxation

Keine Approximation eindeutig besser.

Effizienz der SVM^{cluster}

SVM^{cluster} mit Approximation für die Geschwindigkeit die beste Wahl.

Zusammenfassung

SVM^{cluster} vs. PCC

SVM je nach Material gleich oder besser als PCC.

Optimierung für Verlustfunktionen

Wahl der Verlustfunktion entscheidend für Accurracy, abhängig vom Modell.

Verlust bei argmax$_y H(y)$

Verlust in argmax$_y H(y)$ nicht zwingend erforderlich.

Greedy vs. Relaxation

Keine Approximation eindeutig besser.

Effizienz der SVM^{cluster}

SVM^{cluster} mit Approximation für die Geschwindigkeit die beste Wahl.

Danke für eure Aufmerksamkeit.

Fragen?

Danke für eure Aufmerksamkeit.
Fragen?