Computer-Linguistische Anwendungen

CLA | B.Sc. | LMU

Vorlesung: MaxEnt Klassifikator

Philipp Wicke, PhD
Centrum für Sprach- und Informationsverarbeitung
Ludwig-Maximilians-Universität München
pwicke@cis.lmu.de

Übersicht

- MaxEnt Klassifikator (Einführung)
- Exkurs: Lineare Regression
- MaxEnt: Details
- MaxEnt: Verwendung von MaxEnt mit SciKit-learn
- Support Vector Machine (SVM)

MaxEnt Klassifikator (Einführung)

Einführung in den Max Entropy (MaxEnt) Klassifikator

- **Generative Modelle** wie z.B. Naive Bayes schätzen die gemeinsame Wahrscheinlichkeit von Merkmalen und Label P(x, y)
  → Die Vorhersage des Labels gegeben, die Merkmale P(y|x) stützen sich dann auf eine (gute?) Schätzung von P(x)

- **Diskriminative Modelle** optimieren die Vorhersage des Labels (gegeben die Merkmale) direkt
  - Perzeptron
  - Support Vector Machine (SVM)
  - Maximum Entropy Klassifikator

- Wahrscheinlichkeitsbasierte diskriminative Modelle schätzen P(y|x) direkt
  → Schätzung von P(X) nicht nötig

Einführung in den MaxEnt Klassifikator

- Oft liefern diskriminative Modelle bessere Ergebnisse als generative Modelle

- Word sense disambiguation, (Klein and Manning, 2002)
  - Vorhersage der Bedeutung eines Wortes aus Merkmalen eines konkreten Wortvorkommens
  - Ergebnisse mit genau gleichen Kontextmerkmalen:
  - Naive Bayes: 73.6 % Accuracy
  - MaxEnt: 76.1 % Accuracy

- Text classification mit dem Reuters Corpus, (Zhang and Oles, 2001)
  - Klassifizierung von Finanznachrichten in 90 Kategorien (*gas, housing, interest, ...*)
  - Merkmale sind die Wortvorkommen
  - Naive Bayes: 77.0 % F-score
  - MaxEnt: 86.4 % F-score
  - Regularisierung ("*Smoothing*" für diskriminative Modelle) ist hier wichtig!

Was ist ein MaxEnt Klassifikator?

- ... mehrere lineare Regressionen, die in eine Wahrscheinlichkeitsverteilung (class | features) umgewandelt wurden.
- ... ein Perzeptron, bei dem nicht nur das Vorzeichen wichtig ist, sondern auch die Wahrscheinlichkeit berechnet wird.
- ... einer der grundlegendsten Bausteine des Machine-Learning!
- Andere Namen:
  Logistische Regression, Logit-Modell, log-lineares Modell

Woher kommt der Name MaxEnt?
Das Ergebnis ist aus allen möglichen Modellen, welche die Likelihood maximieren, das Modell mit der größten Entropie. (Entropie: Die Entropie einer Zufallsvariable ist der durchschnittliche Gehalt von 'Information', 'Überraschung' oder 'Unsicherheit', der in den potenziellen Ergebnissen der Variable inhärent ist.)

Was ist ein MaxEnt Klassifikator?

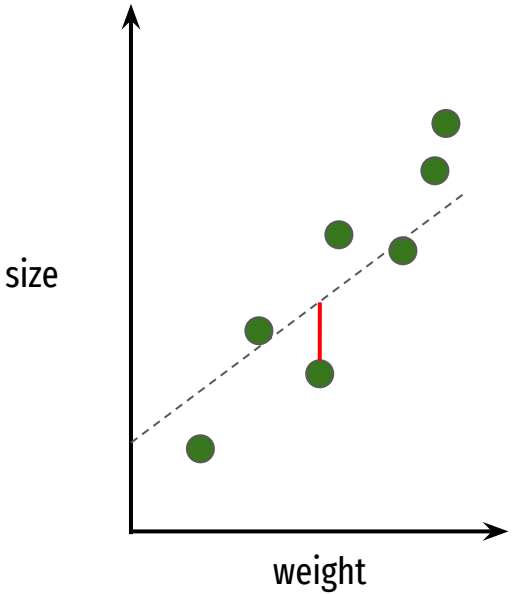
- **Erster Schritt**: Vorhersage eines Scores z_i für jede mögliche Ausgabe-Kategorie i
  - Jeder Merkmalswert wird mit einem entsprechenden Gewicht multipliziert
  - Scores liegen in $]-\infty;\infty[$
  - Die Bedeutung der Ausgabe-Kategorie hängt von der Aufgabe ab, z.B. {GOOD_EMAIL,SPAM} oder {Noun, Verb, Adj, ...} oder {positive, negative, neutral}

- **Dann**: Re-Skalierung and Normalisierung
  - Wahrscheinlichkeit (WK) für jede Kategorie muss positiv sein
  - Summer der WKen aller Kategorien muss 1 ergeben
  - → Exponenzieren und Normalisieren von z

- Lernen der Merkmals-Gewichte: Maximieren der WK der Labels des gesamten Trainings-Datensatzes (*maximum likelihood estimation*)

Exkurs: Lineare Regression

Exkurs: Lineare Regression



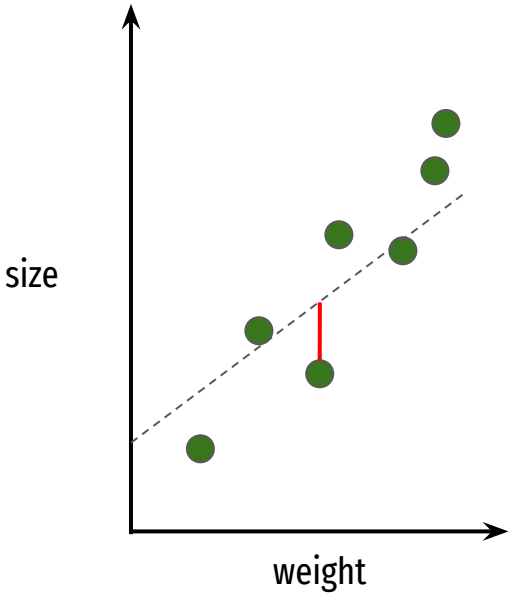
R^2, das Maß der Bestimmtheit, verwendet die Quadratsummenzerlegung, welche die gesamte Quadratsumme in zwei Teile zerlegt: die Quadratsumme, die durch das Regressionsmodell erklärt wird, und die Residuenquadratsumme.

$$R^2 \equiv \frac{SQE}{SQT} = \frac{\sum_i (\hat{y}_i - \bar{y})^2}{\sum_i (y_i - \bar{y})^2},$$

Die Streuung der Schätzwerte $\hat{y}_i$ um ihren Mittelwert $\bar{y}$ kann durch **SQE** gemessen werden. Die Streuung der Messwerte y_i um den Gesamtmittel $\bar{y}$ durch **SQT** gemessen werden.

(**S**umme der **Q**uadrate der **E**rklärten Abweichungen)
(**S**umme der **Q**uadrate **T**otal)

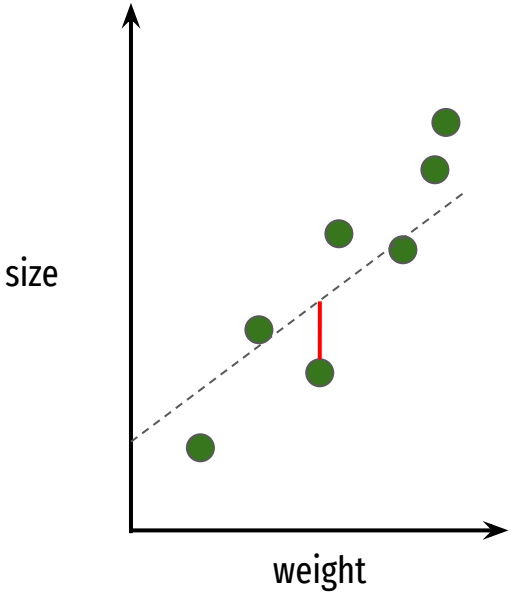
Exkurs: Lineare Regression



"Einfach":
Man berechnet die Quadratische Abweichung der geschätzten Werte vom Mittelwert und die Quadratische Abweichung der tatsächlichen Werte vom Mittelwert. Der Quotient gibt ein Maß der Korrelation an:

$$R^2 \equiv \frac{SQE}{SQT} = \frac{\sum_i (\hat{y}_i - \bar{y})^2}{\sum_i (y_i - \bar{y})^2},$$

Exkurs: Lineare Regression



Mit R^2 lässt sich die Korrelation zwischen *size* und *weighr* bestimmen.

Mit der Berechnung des p-Wertes lässt sich feststellen, ob die Korrelation statistisch signifikant ist.

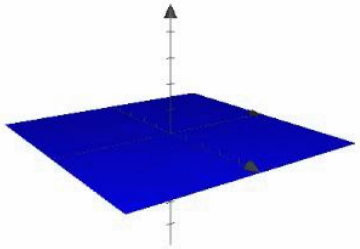
Mit der Korrelations-Funktion lassen sich *size* aus *weight* voraussagen.

Lineare Regression

- *a linear model = a model that assumes a linear relationship between the input variables (x) and the single output variable (y). This implies that (y) can be calculated from a linear combination of the input (x)*

- **Lineare Funktion**: weist jeder input Variable einen Koeffizienten, i.e. Skalierfaktor/Gewicht. Ein zusätzlicher Koeffizient (Intercept/Bias Koeffizient) gibt der Linie einen zusätzlichen Freiheitsgrad (degree of freedom – die Linie wird nach oben oder nach unten in einem 2D plot geschoben):

  $y = b0 + b1 * x_j + b2 * x_i$



$$\hat{y} = \vec{w}^T \vec{x} = \sum_{j=1}^{m} w_j x_j$$

Ziel: Gewichte w_j bestimmen für die der Vorhersagefehler möglichst gering ist.

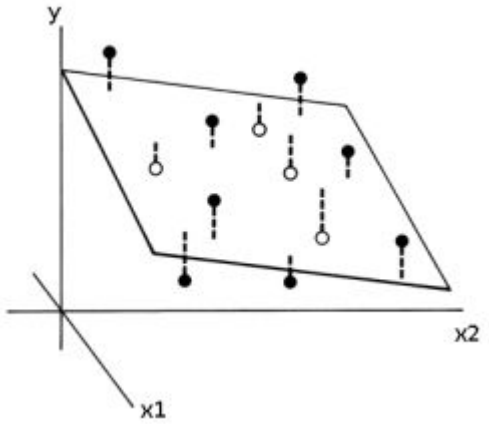
Lineare Regression

- Beispiel:
  Levitt and Dubner (2005) zeigen dass Immobilienpreise z.Z.
  aus dem Vorkommen bestimmter Wortarten (und weiterer Merkmale)
  vorausgesagt werden können

  - “fantastic”, “cute” or “charming”) niedrigerer Preis
  - “maple”, “granite”) höherer Preis

  - Quadratmeter-Preis

  = w0 + w1 · Frequenz_Adjektive + w2 · Zinsrate + w3 · Unverkaufte



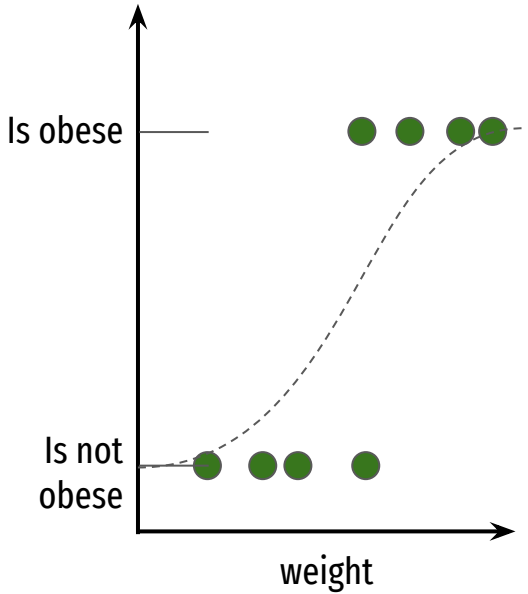
https://commons.wikimedia.org/wiki/File:2d_multiple_linear_regression.gif

Weiter zur Logistic Regression

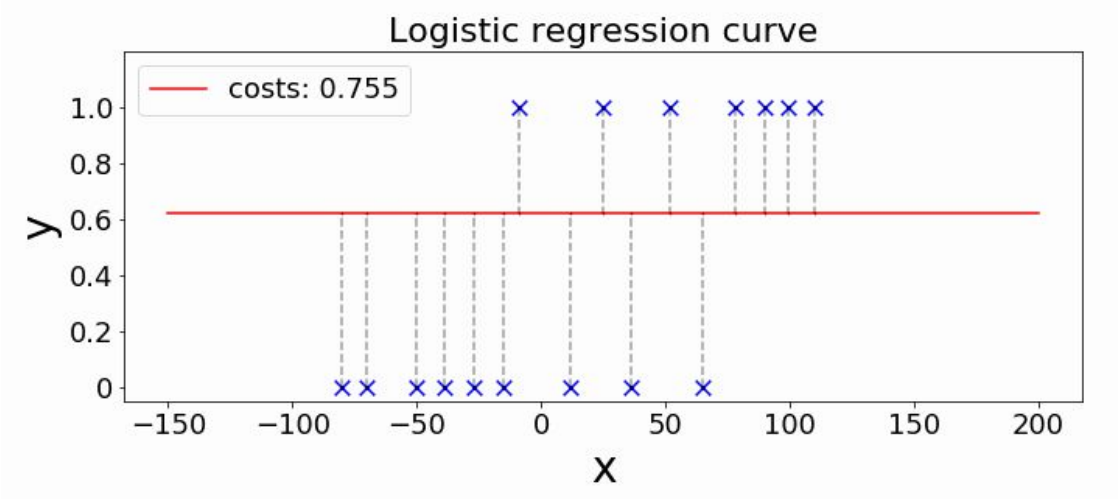
(MaxEnt)

Logistic Regression

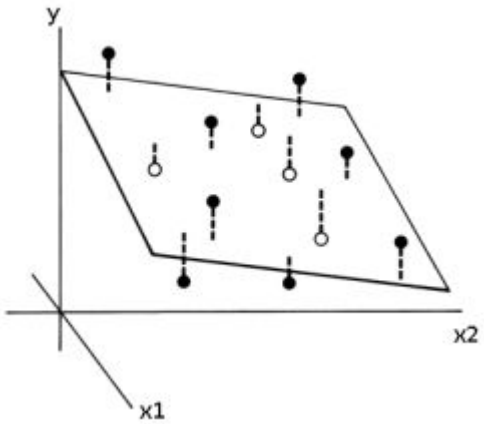
$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}},$$



Anstelle des R^2 Wertes, welcher auf mittlerer quadratischer Abweichung basiert, wird hier stattdessen die **maximum likelihood** berechnet um die Optimierung vorzunehmen.



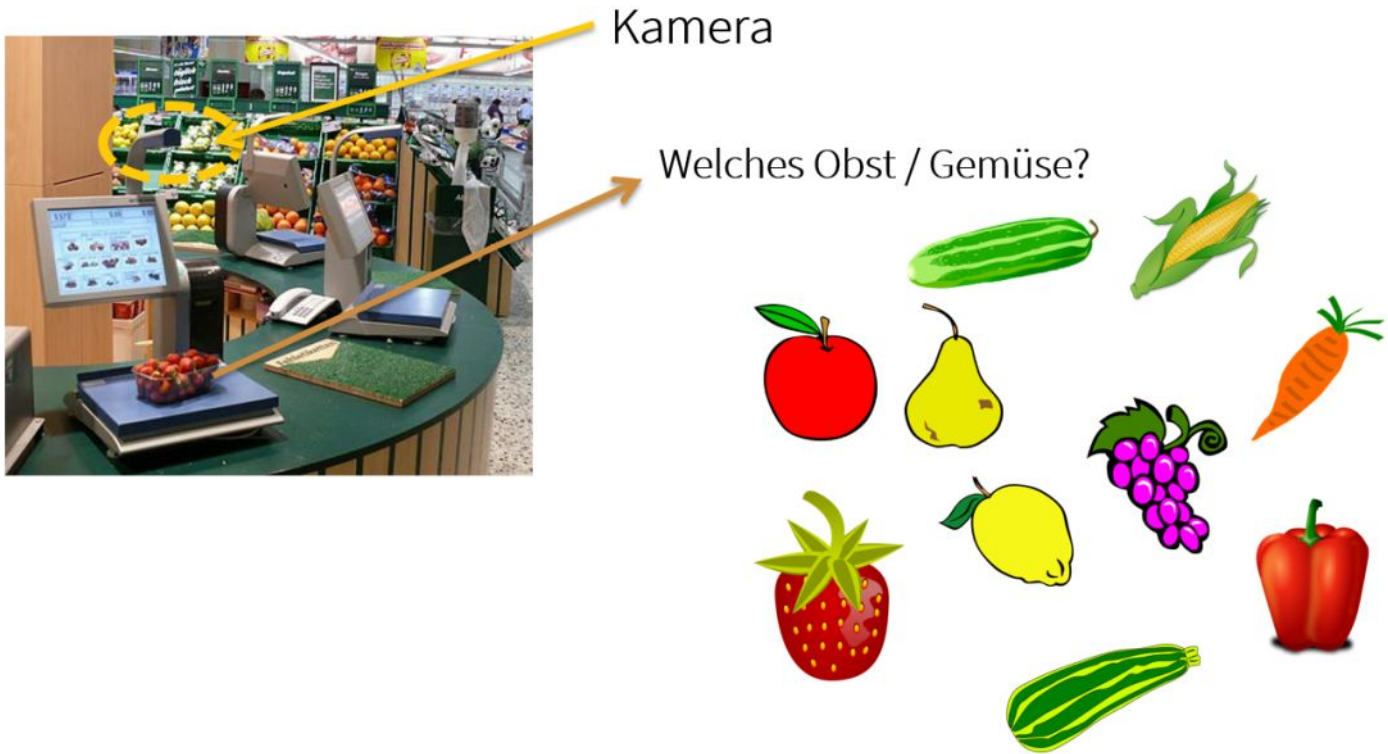
Zurück zu MaxEnt



- Im Maximum Entropy Modell wollen wir Regression zur Klassifikation verwenden
- Wenn *k* Klassen vorherzusagen sind, sagen wir durch Regression einen Score pro Klasse voraus.

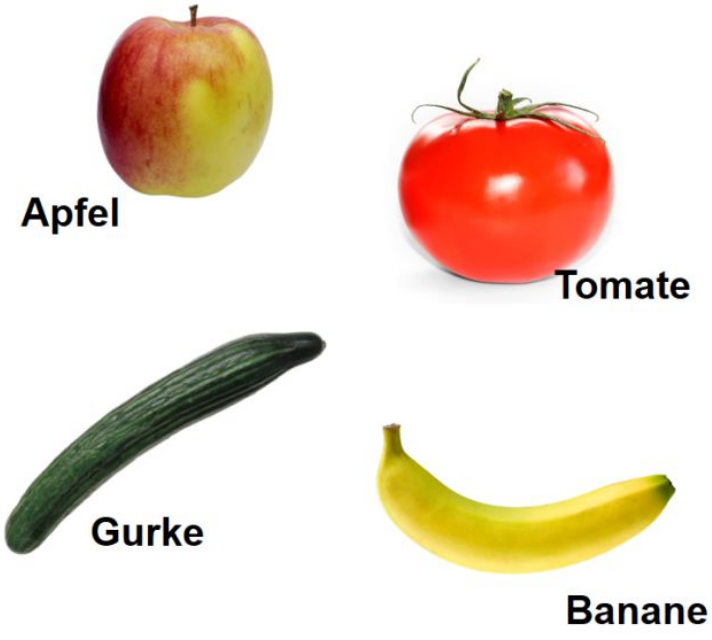
https://commons.wikimedia.org/wiki/File:2d_multiple_linear_regression.gif

Beispiel: Klassifikation von Gemüse



Beispiel: Klassifikation von Gemüse

Ausgabeklassen:



Merkmale:

rot gelb

grün

rund

lang

Beispiel: Tomaten-Score

Analog zum Perzeptron-Score
Welches ist die Tomate, $x^{(1)}$ oder $x^{(2)}$?

$$\vec{w}_{\text{tomate}} = \begin{bmatrix} \\ \\ \\ \\ \\ \end{bmatrix} \begin{matrix} \text{rot} \\ \text{gelb} \\ \text{grün} \\ \text{rund} \\ \text{lang} \end{matrix} \quad \vec{x}^{(1)} = \begin{bmatrix} 0.9 \\ 0 \\ 0.1 \\ 0.9 \\ 0.1 \end{bmatrix} \quad \vec{x}^{(2)} = \begin{bmatrix} 0 \\ 0.8 \\ 0.2 \\ 0 \\ 1 \end{bmatrix}$$

Gute Gewichte für w_{tomate}?
$w^T x^{(1)} =$
$w^T x^{(2)} =$

Apfel-, Bananen-, und Gurkenklassifikator

$$\vec{w}_{\text{apfel}} = \begin{bmatrix} \\ \\ \\ \\ \\ \end{bmatrix} \begin{matrix} \text{rot} \\ \text{gelb} \\ \text{grün} \\ \text{rund} \\ \text{lang} \end{matrix} \quad \vec{w}_{\text{banane}} = \begin{bmatrix} \\ \\ \\ \\ \\ \end{bmatrix} \quad \vec{w}_{\text{gurke}} = \begin{bmatrix} \\ \\ \\ \\ \\ \end{bmatrix}$$

Apfel-, Bananen-, und Gurkenklassifikator

$$\vec{w}_{\text{apfel}} = \begin{bmatrix} 0{,}5 \\ 0{,}5 \\ 0{,}5 \\ 1 \\ -2 \end{bmatrix} \begin{matrix} \text{rot} \\ \text{gelb} \\ \text{grün} \\ \text{rund} \\ \text{lang} \end{matrix} \quad \vec{w}_{\text{banane}} = \begin{bmatrix} -2 \\ 1 \\ 0.5 \\ -1 \\ 1 \end{bmatrix} \quad \vec{w}_{\text{gurke}} = \begin{bmatrix} -1 \\ -1 \\ 1 \\ -1 \\ 1.5 \end{bmatrix}$$

Apfel-, Bananen-, und Gurkenklassifikator

- Wieviele Merkmalsgewichte gibt es?
- Wie schreiben wir die Multiplikation der Design-Matrix mit den Merkmalsgewichten?

Apfel-, Bananen-, und Gurkenklassifikator

- Wieviele Merkmalsgewichte gibt es?

  Ein Gewicht pro Kombination aus Merkmal und Kategorie:

  5 * 4 = 20 Gemüse-Gewichte

$$\vec{W} = \begin{bmatrix} \vec{w}_{\text{tomate}}^{T} \\ \vec{w}_{\text{apfel}}^{T} \\ \vec{w}_{\text{banane}}^{T} \\ \vec{w}_{\text{gurke}}^{T} \end{bmatrix}$$

Gewichte für MaxEnt-Klassifikator

- Wieviele Merkmalsgewichte gibt es?
- Wie schreiben wir die Multiplikation der Design-Matrix mit den Merkmalsgewichten?

  Bei *n* Instanzen, *m* Merkmalen und *k* Klassen, hat die Design-Matrix **X** die Größe *n* × *m* und die Größe der Gewichts-Matrix W wird mit *k* × *m* festgelegt. Die Matrix mit Vorhersage-Scores für alle Instanzen ist $\vec{Z} = \vec{X}\vec{W}^T$

$$\vec{X} = \begin{bmatrix} \vec{x}^{(1)T} \\ \vec{x}^{(2)T} \\ \cdots \\ \vec{x}^{(n)T} \end{bmatrix}$$

Vorhersage-Scores für eine Instanz

k Klassen und m Merkmale

m-dimensionaler Merkmalsvektor für eine Instanz: $\vec{x}$

Gewichts-Matrix $\vec{W}$ mit Größe $k \times m$

$$\vec{z} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_k \end{bmatrix} = \vec{W}\vec{x}$$

(d.h. $\vec{z}$ hat eine Komponente pro Ausgabe-Klasse)

Wahrscheinlichkeiten berechnen

Scores z_i können positiv oder negativ werden

Wenn wir die geschätzte WK einer Ausgabeklasse wissen möchten, sind Scores nur begrenzt nützlich

Ziel: Berechnen der WK-Verteilung $\hat{\vec{y}}$ über alle k Ausgabeklassen (gegeben eine Instanz $\vec{x}$).

$$\hat{\vec{y}} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_k \end{bmatrix} = \begin{bmatrix} P(Y = 1|\vec{x}, \vec{W}) \\ P(Y = 2|\vec{x}, \vec{W}) \\ \vdots \\ P(Y = k|\vec{x}, \vec{W}) \end{bmatrix}$$

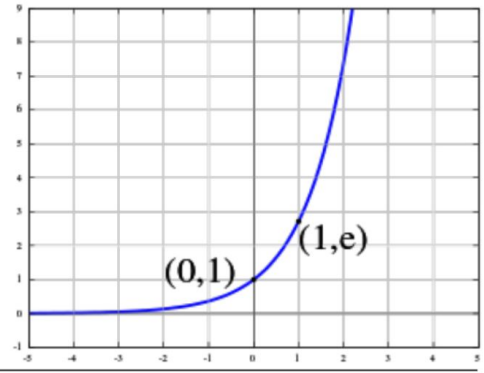
Wahrscheinlichkeiten aus Scores berechnen

Bedingungen für WK-Verteilung

1. **Positiv**
   $\Rightarrow exp(z_i)$ statt z_i
2. **WKen über alle Klassen addieren sich zu 1**

$$\Rightarrow \frac{exp(z_i)}{\sum_{j=1}^{k} exp(z_j)}$$

$exp(x) = e^x \approx 2,72^x$



$$\hat{\vec{y}} = \begin{bmatrix} P(Y=1|\vec{x},\vec{W}) \\ P(Y=2|\vec{x},\vec{W}) \\ \vdots \\ P(Y=k|\vec{x},\vec{W}) \end{bmatrix} = \begin{bmatrix} \frac{\exp(z_1)}{\sum_i \exp(z_i)} \\ \frac{\exp(z_2)}{\sum_i \exp(z_i)} \\ \vdots \\ \frac{\exp(z_k)}{\sum_i \exp(z_i)} \end{bmatrix}$$

Regularisierung (vermeiden von Overfitting)

- Es kann passieren, dass die Trainingsdaten zu genau modelliert werden
- Z.B. wenn die WKen immer ~ 0 oder ~ 1 sind (und die Training-Labels nachbilden).

  → der Klassifikator liefert dann keine guten Vorhersagen für nicht im Training vorkommende Merkmalskombinationen

- **Welche Scores entsprechen solchen extremen WKen?**

- **Welche Gewichte entsprechen solchen Scores?**

Regularisierung (vermeiden von Overfitting)

- **Welche Scores entsprechen solchen extremen WKen?**

  Positive oder negative Scores mit großen Absolutbeträgen
- **Welche Gewichte entsprechen solchen Scores?**

  Positive oder negative Gewichte mit großen Absolutbeträgen
- Lösung: **Kleine** Gewichte in W~ zu erhalten, soll ein weiteres Trainingsziel sein
  - L1-Regularisierung: Absolutbeträge der Gewichte werden zur Fehlerfunktion (negative Log-likelihood) dazu addiert.

    → Ergibt eine **Sparse** Matrix der Merkmalsgewichte (viele Gewichte sind 0) → reduziert Effekt von nicht signifikanten Merkmalen (macht eine Art Feature Selection)
  - L2-Regularisierung: Quadrate der Gewichte werden zur Fehlerfunktion dazu addiert.

    → Ergibt eine Merkmalsgewichts-Matrix, in der die meisten Einträge kleine Zahlen ≠ 0 sind

Wie findet der Klassifikator die besten Gewichte?

Während des Trainings wird für jedes Trainingsbeispiel die Wahrscheinlichkeit des Labels, also $P(Y = y^{(i)}|\vec{x}^{(i)}, \vec{W})$ berechnet.

Das Produkt aller solcher Wahrscheinlichkeiten nennt man die **Likelihood** der Parameter für den Datensatz:

$$Likelihood(\vec{W}) = P(\vec{y}|\vec{X}, \vec{W}) = \prod_{i=1}^{n} P(Y = y^{(i)}|\vec{x}^{(i)}, \vec{W})$$

Es werden die Gewichte $\vec{W}$ gesucht, die die Likelihood maximieren.

Aber warum kein Deep Learning?

nature

Explore content | About the journal | Publish with us | Subscribe

nature > letters > article

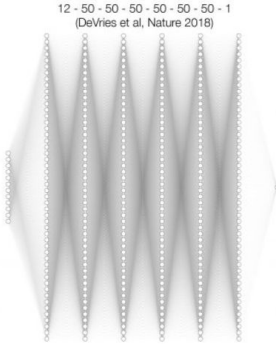
Letter | Published: 29 August 2018

Deep learning of aftershock patterns following large earthquakes

Phoebe M. R. DeVries, Fernanda Viégas, Martin Wattenberg & Brendan J. Meade

Nature **560**, 632–634 (2018) | Cite this article

28k Accesses | **159** Citations | **959** Altmetric | Metrics



- 13,451 parameter DNN (weights and biases)

Image adapted from Eric Vyacheslav

Aber warum kein Deep Learning?

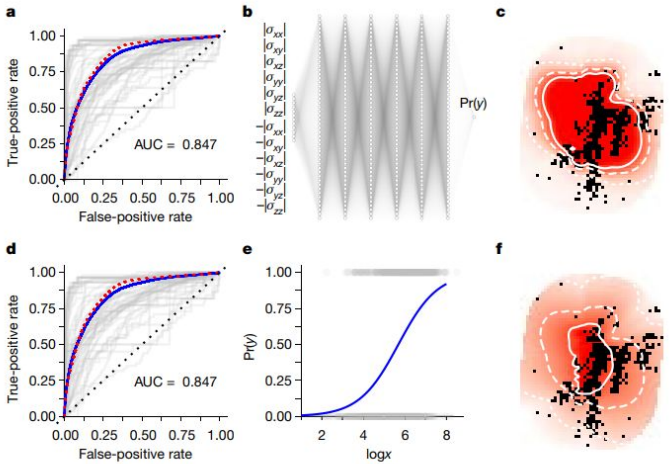


Fig. 1 | Prediction of aftershock spatial patterns based on stress features. a, Receiver operating characteristic (ROC) curves of test data for the DNN of ref. [1]. **b**, DNN topology; plot generated with NN-SVG (http://alexlenail.me/NN-SVG/). **c**, Example of DNN prediction (1999 ChiChi aftershocks). **d**, Test-data ROC curves for logistic regression with the logarithm of the sum of absolute stress components as input. **e**, Logistic regression fit on training data. **f**, Example of logistic regression prediction. For both DNN and logistic regression models, 58 ROC curves are shown, for the 57 mainshocks from the test set (grey) and all combined mainshock–aftershock pairs (blue; see legend of Fig. 2 for red colour explanation). Dotted, dashed and solid curves in **c** and **f** represent $Pr(y) = 0.3$, 0.5 and 0.7, respectively.

nature

Explore content | About the journal | Publish with us

nature > matters arising > article

Matters Arising | Published: 02 October 2019

Reply to: One neuron versus deep learning in aftershock prediction

Brendan J. Meade

Nature **574**, E4 (2019) | Cite this article

4407 Accesses | **3** Citations | **5** Altmetric | Metrics

"we reformulate the 2017 results using two-parameter logistic regression (that is, one neuron) and obtain the same performance as that of the 13,451-parameter DNN"

https://www.nature.com/articles/s41586-019-1583-7.pdf

Image adapted from Eric Vyacheslav

Verwendung

des MaxEnt Klassifikators

MaxEnt in Scikit-Learn

sklearn.linear_model.LogisticRegression

class `sklearn.linear_model.LogisticRegression`(*penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None*) [source]

Logistic Regression (aka logit, MaxEnt) classifier.

In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme if the 'multi_class' option is set to 'ovr', and uses the cross-entropy loss if the 'multi_class' option is set to 'multinomial'. (Currently the 'multinomial' option is supported only by the 'lbfgs', 'sag', 'saga' and 'newton-cg' solvers.)

This class implements regularized logistic regression using the 'liblinear' library, 'newton-cg', 'sag', 'saga' and 'lbfgs' solvers. **Note that regularization is applied by default**. It can handle both dense and sparse input. Use C-ordered arrays or CSR matrices containing 64-bit floats for optimal performance; any other input format will be converted (and copied).

The 'newton-cg', 'sag', and 'lbfgs' solvers support only L2 regularization with primal formulation, or no regularization. The 'liblinear' solver supports both L1 and L2 regularization, with a dual formulation only for the L2 penalty. The Elastic-Net regularization is only supported by the 'saga' solver.

Read more in the User Guide.

Parameters:

penalty : *{'l1', 'l2', 'elasticnet', None}, default='l2'*

Specify the norm of the penalty:

- `None`: no penalty is added;
- `'l2'`: add a L2 penalty term and it is the default choice;
- `'l1'`: add a L1 penalty term;
- `'elasticnet'`: both L1 and L2 penalty terms are added.

MaxEnt in Scikit-Learn

Hyper-Parameter (Training):

- `penalty` : l1 or l2
- `C` : Kehrwert der Regularisierungsstärke (positive float, kleinere Werte definieren stärkere Regularisierung)
- solver, ...

Eingabe (Training):

- X : Design-Matrix (shape: num_instances × features)
- y: Labels (Vektor mit num_instances Integer-Werten, welche die korrekte Klasse anzeigen 0,1,...)

Die resultierende Gewichtsmatrix zeigt, welche Merkmale besonders hohe Gewichte haben

MaxEnt in Scikit-Learn

```
from sklearn import linear_model

#(X, y) = (features matrix, labels)
maxent = linear_model.LogisticRegression(penalty='l2', C=1.0))
maxent.fit(X_train, y_train)
print(maxent.coef_)
# That is a matrix with the shape (n_classes, n_features)
```

Vorhersage und Evaluierung

```
from sklearn.metrics import accuracy_score

# Predicts vector with (integer) labels.
y_predicted = maxent.predict(X_dev)
dev_acc = accuracy_score(y_dev, y_predicted)

# Probability distribution over all possible classes
# Shape: (n_instances, n_classes)
y_probs = maxent.predict_proba(X_dev)
```

... Berechnung der F-Scores für alle Klassen:

```
from sklearn.metrics import f1_score
y_true = [0, 1, 2, 0, 1, 2]
y_pred = [0, 2, 1, 0, 0, 1]
f1_score(y_true, y_pred, average=None)
# average= None | needed for multi-class classification
# returns f1-score for each class
array([ 0.8, 0. , 0. ])
```

average : *{'micro', 'macro', 'samples', 'weighted', 'binary'} or None, default='binary'*

This parameter is required for multiclass/multilabel targets. If `None`, the scores for each class are returned. Otherwise, this determines the type of averaging performed on the data:

`'binary'`:
Only report results for the class specified by `pos_label`. This is applicable only if targets (`y_{true,pred}`) are binary.

`'micro'`:
Calculate metrics globally by counting the total true positives, false negatives and false positives.

`'macro'`:
Calculate metrics for each label, and find their unweighted mean. This does not take label imbalance into account.

Scikit-learn Logistic Regression für die Paraphrasen-Erkennung

Wie erhalten wir **X** und **y**?
→ `DictVectorizer`

Welche Hyperparameter können optimiert werden?
→ Merkmale, Regularisierungsart, Regularisierungsstärke