Supervised Classification: Logistic Regression

CSE354 - Spring 2020
Special Topic in CS
NLP’s practical applications

- Machine translation
- Automatic speech recognition
  - Personalized assistants
  - Auto customer service
- Information Retrieval
  - Web Search
  - Question Answering
- Sentiment Analysis
- Computational Social Science
- Growing day by day

Machine learning:
  - Logistic regression
  - Probabilistic modeling
  - Recurrent Neural Networks
  - Transformers

Algorithms, e.g.:
  - Graph analytics
  - Dynamic programming

Data science
  - Hypothesis testing
NLP’s practical applications

- Machine translation
- Automatic speech recognition
  - Personalized assistants
  - Auto customer service
- Information Retrieval
  - Web Search
  - Question Answering
- Sentiment Analysis
- Computational Social Science
- Growing day by day

- Machine learning:
  - Logistic regression
  - Probabilistic modeling
  - Recurrent Neural Networks
  - Transformers

- Algorithms, e.g.:
  - Graph analytics
  - Dynamic programming

- Data science
  - Hypothesis testing
Topics we will cover

- Supervised Classification
- Goal of logistic regression
- The “loss function” — what logistic regression tries to optimize
- Adding Multiple Features
- Training and Test Sets
- Overfitting; Role of Regularization
Supervised Classification

\( X \) - features of \( N \) observations (i.e. words)

\( Y \) - class of each of \( N \) observations

**GOAL:** Produce a *model* that outputs the most likely class \( y_i \), given features \( x_i \).

\[ f(X) = Y \]
Supervised Classification

**X** - features of N observations (i.e. words)

**Y** - class of each of N observations

**GOAL:** Produce a *model* that outputs the most likely class $y_i$, given features $x_i$.

$$f(X) = Y$$

<table>
<thead>
<tr>
<th>$i$</th>
<th>$X$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0.25</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0.75</td>
<td>1</td>
</tr>
</tbody>
</table>
Supervised Classification

$X$ - features of N observations (i.e. words)

$Y$ - class of each of N observations

GOAL: Produce a model that outputs the most likely class $y_i$, given features $x_i$.

$$f(X) = Y$$

Some function or rules to go from $X$ to $Y$, as close as possible.
Supervised Classification

*Supervised* Machine Learning: Build a model with examples of outcomes (i.e. $Y$) that one is trying to predict.

*Classification*: The outcome ($Y$) is a discrete class (e.g. \{noun, verb, adjective, adverb\}; \{positive sentiment, negative sentiment\}).
Logistic Regression

Binary classification goal: Build a model that can estimate $P(A=1|B=\text{?})$

i.e. given B, yield (or “predict”) the probability that A=1
Logistic Regression

Binary classification goal: Build a “model” that can estimate $P(A=1|B=?)$

i.e. given $B$, yield (or “predict”) the probability that $A=1$

In machine learning, tradition to use $Y$ for the variable being predicted and $X$ for the features use to make the prediction.
Logistic Regression

Binary classification goal: Build a “model” that can estimate $P(Y=1|X=?)$

i.e. given $X$, yield (or “predict”) the probability that $Y=1$

In machine learning, tradition is to use $Y$ for the variable being predicted and $X$ for the features used to make the prediction.
Logistic Regression

Binary classification goal: Build a “model” that can estimate $P(Y=1|X=?)$

i.e. given $X$, yield (or “predict”) the probability that $Y=1$

In machine learning, tradition is to use $Y$ for the variable being predicted and $X$ for the features used to make the prediction.

Example:  
$Y$: 1 if target is verb, 0 otherwise;  
$X$: 1 if “was” occurs before target; 0 otherwise

$I$ was *reading* for NLP.  
$We$ were *fine*.  
$I$ am *good*.

*The* cat was *very* happy.  
*We* enjoyed the *reading* material.  
*I* was *good*. 
Logistic Regression

Binary classification goal: Build a “model” that can estimate $P(Y=1|X=?)$

i.e. given $X$, yield (or “predict”) the probability that $Y=1$

In machine learning, tradition is to use $Y$ for the variable being predicted and $X$ for the features used to make the prediction.

Example: $Y$: 1 if target is verb, 0 otherwise; $X$: 1 if “was” occurs before target; 0 otherwise

*I was reading* for NLP.  
*We were fine.*  
*I am good.*

*The cat was very happy.*  
*We enjoyed the reading material.*  
*I was good.*
Logistic Regression

Example:  
- $Y$: 1 if target is a part of a proper noun, 0 otherwise;
- $X$: number of capital letters in target and surrounding words.

*They attend Stony Brook University.*  
*Next to the brook Gandalf lay thinking.*

*The trail was very stony.*  
*Her degree is from SUNY Stony Brook.*

*The Taylor Series was first described by Brook Taylor, the mathematician.*
Logistic Regression

Example: $Y$: 1 if target is a part of a proper noun, 0 otherwise;
$X$: number of capital letters in target and surrounding words.

They attend Stony Brook University. Next to the brook Gandalf lay thinking.

The trail was very stony. Her degree is from SUNY Stony Brook.

The Taylor Series was first described by Brook Taylor, the mathematician.
Logistic Regression

Example: \( Y \): 1 if target is a part of a proper noun, 0 otherwise; 
\( X \): number of capital letters in target and surrounding words.

\[
\begin{array}{c|c}
 x & y \\
\hline
 2 & 1 \\
 1 & 0 \\
 0 & 0 \\
 6 & 1 \\
 2 & 1 \\
\end{array}
\]

They attend Stony Brook University. Next to the brook Gandalf lay thinking.

The trail was very stony. Her degree is from SUNY Stony Brook.

The Taylor Series was first described by Brook Taylor, the mathematician.
Logistic Regression

Example: 

Y: 1 if target is a part of a proper noun, 0 otherwise;  
X: number of capital letters in target and surrounding words.

They attend Stony Brook University. Next to the brook Gandalf lay thinking.

The trail was very stony. Her degree is from SUNY Stony Brook.

The Taylor Series was first described by Brook Taylor, the mathematician.

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Logistic Regression

Example: \( Y: 1 \) if target is a part of a proper noun, 0 otherwise;
\( X: \) number of capital letters in target and surrounding words.

They attend Stony Brook University.  Next to the brook Gandalf lay thinking.

The trail was very stony.  Her degree is from SUNY Stony Brook.

The Taylor Series was first described by Brook Taylor, the mathematician.
Logistic Regression

Example:  
\[ Y: 1 \text{ if target is a part of a proper noun, } 0 \text{ otherwise; } \]
\[ X: \text{ number of capital letters in target and surrounding words.} \]

They attend Stony Brook University.  
Next to the brook Gandalf lay thinking.

The trail was very stony.  
Her degree is from SUNY Stony Brook.

The Taylor Series was first described by Brook Taylor, the mathematician.

They attend Binghamton.

<table>
<thead>
<tr>
<th>( x )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Logistic Regression

Example: \( Y \): 1 if target is a part of a proper noun, 0 otherwise; \( X \): number of capital letters in target and surrounding words.

\[
\begin{array}{c|c}
 x & y \\
\hline
 2 & 1 \\
 1 & 0 \\
 0 & 0 \\
 6 & 1 \\
 2 & 1 \\
 1 & 1 \\
\end{array}
\]

They attend Stony Brook University. Next to the brook Gandalf lay thinking.

The trail was very stony. Her degree is from SUNY Stony Brook.

The Taylor Series was first described by Brook Taylor, the mathematician.

They attend Binghamton.
Logistic Regression

Example:  
Y: 1 if target is a part of a proper noun, 0 otherwise; 
X: number of capital letters in target and surrounding words.

```
Out[43]: [<matplotlib.lines.Line2D at 0x116e68d68>]

In [78]: 1 -b_0/b_1
Out[78]: 0.5824799517820446

In [28]: 1 logisticRegr.predict(x)
Out[28]: array([[1, 1, 0, 1, 1]])
```

```
Out[80]: [<matplotlib.lines.Line2D at 0x112a60f160>]

In [81]: 1 -b2_0/b2_1
Out[81]: 0.3108939388058134

In [82]: 1 logisticRegr2.predict(x2)
Out[82]: array([[1, 1, 0, 1, 1]])
```
Logistic Regression

Example:  

\[ Y: \begin{cases} 1 & \text{if target is a part of a proper noun,} \\ 0 & \text{otherwise;} \end{cases} \]

\[ X: \text{number of capital letters in target and surrounding words.} \]

In [78]: 1 -b_0/b_1
Out[78]: 0.5824799517820446

In [28]: logisticRegr.predict(x)
Out[28]: array([1, 1, 0, 1, 1])

In [81]: -b2_0/b2_1
Out[81]: 0.3108930938058134

In [82]: logisticRegr2.predict(x2)
Out[82]: array([1, 1, 0, 1, 1])
Logistic Regression

Example:   \( Y: 1 \) if target is a part of a proper noun, 0 otherwise; 
\( X: \) number of capital letters in target and surrounding words.

\[
x = \begin{pmatrix} 0 & 1 & 1 \end{pmatrix}
y = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}
\]

optimal \( b_0, b_1 \) changed!

```python
In [43]: [matplotlib.lines.Line2D at 0x116e6d60]
Out[43]:

In [80]: [matplotlib.lines.Line2D at 0x11660f160]
Out[80]:

In [78]: 1 -b_0/b_1
Out[78]: 0.5824799517820446

In [78]: 1 logisticRegr.predict(x)
Out[28]: array([1, 1, 0, 1, 1])

In [81]: 1 -b_0/b2_1
Out[81]: 0.31089309388058134

In [81]: 1 logisticRegr2.predict(x2)
Out[82]: array([1, 1, 0, 1, 1])
```

\[
\begin{pmatrix} 1 & 1 & 0 & 1 & 1 \end{pmatrix}
\]
Logistic Regression on a single feature ($x$)

$Y_i \in \{0, 1\};$ $X$ is a **single value** and can be anything numeric.

$$p_i \equiv P(Y_i = 1 | X_i = x) = \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}}$$
Logistic Regression on a single feature ($x$)

$Y_i \in \{0, 1\}; \ X$ is a **single value** and can be anything numeric.

\[
p_i \equiv P(Y_i = 1 \mid X_i = x) = \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}} = \frac{1}{1 + e^{-(\beta_0 + \sum_{j=1}^{m} \beta_j x_{ij})}}
\]
Logistic Regression on a single feature \((x)\)

\(Y_i \in \{0, 1\}; X\) can be anything numeric.

\[
p_i \equiv P(Y_i = 1 | X_i = x) = \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}}
\]

The goal of this function is to: take in the variable \(x\) and return a probability that \(Y\) is 1.
Logistic Regression on a single feature ($x$)

$Y_i \in \{0, 1\}; X$ can be anything numeric.

$$p_i \equiv P(Y_i = 1 \mid X_i = x) = \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}}$$

The goal of this function is to: take in the variable $x$ and return a probability that $Y$ is 1.

Note that there are only three variables on the right: $X_i$, $\beta_0$, $\beta_1$.
Logistic Regression on a single feature \((x)\)

\(Y_i \in \{0, 1\}; \ X\) can be anything numeric.

\[
p_i \equiv P(Y_i = 1 | X_i = x) = \frac{e^{B_0 + B_1 x_i}}{1 + e^{B_0 + B_1 x_i}}
\]

The goal of this function is to: \text{take in the variable } x \text{ and return a probability that } Y \text{ is 1.}

Note that there are only three variables on the right: \(X_i, B_0, B_1\)

\(X\) is given. \(B_0\) and \(B_1\) must be learned.
Logistic Regression on a single feature ($x$)

$Y_i \in \{0, 1\}; X$ can be anything numeric.

$$p_i \equiv P(Y_i = 1 \mid X_i = x) = \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}}$$

HOW? Essentially, try different $\beta_0$ and $\beta_1$ values until “best fit” to the training data (example $X$ and $Y$).

$X$ is given. $\beta_0$ and $\beta_1$ must be learned.
“best fit” : whatever maximizes the likelihood function:

\[
L(\beta_0, \beta_1 | X, Y) = \prod_{i=1}^{n} p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}
\]

\[
p_i \equiv P(Y_i = 1 | X_i = x) = \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}}
\]

HOW? Essentially, try different \( B_0 \) and \( B_1 \) values until “best fit” to the training data (example \( X \) and \( Y \)).

\( X \) is given. \( B_0 \) and \( B_1 \) must be learned.
X can be multiple features

Often we want to make a classification based on multiple features:

- Number of capital letters surrounding: integer
- Begins with capital letter: \{0, 1\}
- Preceded by “the”? \{0, 1\}
X can be multiple features

Often we want to make a classification based on multiple features:

- Number of capital letters surrounding:
  - integer
- Begins with capital letter: \{0, 1\}
- Preceded by "the"? \{0, 1\}

Y-axis is Y (i.e. 1 or 0)

To make room for multiple Xs, let’s get rid of y-axis. Instead, show decision point.
X can be multiple features

Often we want to make a classification based on multiple features:

- Number of capital letters surrounding: integer
- Begins with capital letter: \{0, 1\}
- Preceded by “the”? \{0, 1\}

We're learning a linear (i.e. flat) separating hyperplane, but fitting it to a logit outcome.
X can be multiple features

Often we want to make a classification based on multiple features:

- Number of capital letters surrounding: integer
- Begins with capital letter: \{0, 1\}
- Preceded by “the”? \{0, 1\}
X can be multiple features

Often we want to make a classification based on multiple features:

- Number of capital letters surrounding: integer
- Begins with capital letter: \{0, 1\}
- Preceded by “the”? \{0, 1\}

We’re learning a linear (i.e. flat) separating hyperplane, but fitting it to a logit outcome.

(https://www.linkedin.com/pulse/predicting-outcomes-probabilities-logistic-regression-konstantinidis/)
Logistic Regression

$Y_i \in \{0, 1\}; \ X$ can be anything numeric.

$p_i \equiv P(Y_i = 1 | X_i = x) = \frac{e^{\beta_0 + \sum_{j=1}^{m} \beta_j x_{ij}}}{1 + e^{\beta_0 + \sum_{j=1}^{m} \beta_j x_{ij}}}$

$logit(p_i) = log \left( \frac{p_i}{1 - p_i} \right) = \beta_0 + \sum_{j=1}^{m} \beta_j x_{ij} = 0$

We’re still learning a linear separating hyperplane, but fitting it to a logit outcome.

(https://www.linkedin.com/pulse/predicting-outcomes-probabilities-logistic-regression-konstantinidis/)
Logistic Regression

Example: Y: 1 if target is a part of a proper noun, 0 otherwise; X: number of capital letters in target and surrounding words.

They attend Stony Brook University. Next to the brook Gandalf lay thinking.

The trail was very stony. Her degree is from SUNY Stony Brook.

The Taylor Series was first described by Brook Taylor, the mathematician.

They attend Binghamton.

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Logistic Regression

Example: $Y$: 1 if target is a part of a proper noun, 0 otherwise;
$X_1$: number of capital letters in target and surrounding words.

Let’s add a feature! $X_2$: does the target word start with a capital letter?

They attend Stony Brook University. Next to the brook Gandalf lay thinking.

The trail was very stony. Her degree is from SUNY Stony Brook.

The Taylor Series was first described by Brook Taylor, the mathematician.

They attend Binghamton.
Machine Learning: How to setup data

Data

Model

training
### Machine Learning: How to setup data

<table>
<thead>
<tr>
<th>$i$</th>
<th>$X$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0.25</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0.75</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>N</td>
<td>0.35</td>
<td>0</td>
</tr>
</tbody>
</table>

---

**Model**

- Training data

---
**Machine Learning: How to setup data**

**“Corpus”**

Raw data: sequences of characters

<table>
<thead>
<tr>
<th>i</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0.25</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0.75</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>0.35</td>
<td>1</td>
</tr>
</tbody>
</table>
Machine Learning: How to setup data

Feature Extraction

--pull out observations and feature vector per observation.

“Corpus”
raw data: sequences of characters

<table>
<thead>
<tr>
<th>i</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0.25</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0.75</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>N</td>
<td>0.35</td>
<td>1</td>
</tr>
</tbody>
</table>
Machine Learning: How to setup data

Feature Extraction

- pull out *observations* and *feature vector* per observation.

*e.g.:* words, sentences, documents, users.

<table>
<thead>
<tr>
<th>i</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0.25</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0.75</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>N</td>
<td>0.35</td>
<td>1</td>
</tr>
</tbody>
</table>
Machine Learning: How to setup data

Feature Extraction

--pull out observations and feature vector per observation.

e.g.: words, sentences, documents, users.

row of features; e.g.

- number of capital letters
- whether "I" was mentioned or not

```
<table>
<thead>
<tr>
<th></th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0 0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0.5 1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1.0 1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0.25 0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0.75 0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>0.35 1</td>
<td>0</td>
</tr>
</tbody>
</table>
```
Machine Learning: How to setup data

Feature Extraction

--pull out *observations* and *feature vector* per observation.

e.g.: words, sentences, documents, users.

row of features; e.g.

➔ number of capital letters
➔ whether “I” was mentioned or not
➔ *k* features indicating whether *k* words were mentioned or not

<table>
<thead>
<tr>
<th>i</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0.25</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0.75</td>
<td>0</td>
</tr>
<tr>
<td>N</td>
<td>0.35</td>
<td>0</td>
</tr>
</tbody>
</table>
Machine Learning: How to setup data

Feature Extraction

Multi-hot Encoding
- Each word gets an index in the vector
- "Corpus" → 1 if present; 0 if not

raw data: sequences of characters
- number of capital letters
- whether "I" was mentioned or not
- $k$ features indicating whether $k$ words were mentioned or not

Data

X    Y
Machine Learning: How to setup data

Feature Extraction

Multi-hot Encoding

- Each word gets an index in the vector
- “Corpus” raw data: sequences of characters
- Each word is encoded as a binary vector:
  - 1 if present; 0 if not

Feature example: is word present in document?

The book was interesting so I was happy.

Data

- $X$: features, e.g.
  - number of capital letters
  - whether “I” was mentioned or not
  - $k$ features indicating whether $k$ words were mentioned or not
Machine Learning: How to setup data

Feature Extraction

Multi-hot Encoding
- Each word gets an index in the vector
- “Corpus”
- 1 if present; 0 if not

Feature example: is word present in document?

The book was interesting so I was happy.

[0, 1, 1, 0, 1, …, 1, 0, 1, 1, 0, 1, …, 1]^k

k features indicating whether k words were mentioned or not
Machine Learning: How to setup data

Feature Extraction

Multi-hot Encoding
- Each word gets an index in the vector
- "1" if present; "0" if not

Feature example: is word present in document

**The book was interesting so I was happy.**

\[ [0, 1, 1, 0, 1, ..., 1, 0, 1, 1, 0, 1, ..., 1^k] \]

raw data: sequences of characters

X

Y

Data

sad
Machine Learning: How to setup data

Feature Extraction

**Multi-hot Encoding**
- Each word gets an index in the vector
- "Corpus" raw data: sequences of characters
- 1 if present; 0 if not

Feature example: is previous word “the”? *The book was interesting so I was happy.*

Data

\[
\begin{bmatrix}
0, & 1, & 1, & 0, & 1, & \ldots, & 1, & 0, & 1, & 1, & 0, & 1, & \ldots, & 1
\end{bmatrix}^k
\]

\(k\) features indicating whether \(k\) words were mentioned or not
Machine Learning: How to setup data

Feature Extraction

Multi-hot Encoding
- Each word gets an index in the vector
- "Campus" → 1 if present; 0 if not

Feature example: is previous word "the"?

Raw data: The book was interesting so I was happy.

Data structure: $[0, 1, 1, 0, 1, \ldots, 1, 0, 1, 1, 0, 1, \ldots, 1]^k$

$k$ features indicating whether $k$ words were mentioned or not.
Machine Learning: How to setup data

Feature Extraction

One-hot Encoding
- Each word gets an index in the vector
- "Computer"
- All indices 0 except present word:
- Feature example: is previous word "the"?

Data
- raw data: sequences of characters
- "The book was interesting so I was happy."
- $[0, 1, 0, 0, 0, 0, ..., 0, 0, 0, 0, 0, 0, 0, 0, 0]^{k}$
- $k$ features indicating whether $k$ words were mentioned or not
Machine Learning: How to setup data

Feature Extraction

One-hot Encoding
- Each word gets an index in the vector
- "Corpus"
- All indices 0 except present word:
- Feature example: which is previous word?
- The book was interesting so I was happy.
- \[ X \]
- \[ Y \]
- raw data: sequences of characters
- \[ [0, 1, 0, 0, 0, \ldots, 0, 0, 0, 0, 0, 0, 0, 0, \ldots, 0] \]
- \[ [0, 0, 1, 0, 0, \ldots, 0, 0, 0, 0, 0, 0, 0, 0, \ldots, 0] \]
Machine Learning: How to setup data

Feature Extraction

One-hot Encoding
- Each word gets an index in the vector
- All indices 0 except present word

Feature example: which is previous word?

raw data: The book was interesting so I was happy

Data

\[
\begin{align*}
[X & Y] \\
[0, 1, 0, 0, 0, \ldots, 0, 0, 0, 0, 0, 0, \ldots, 0]_k \\
[0, 0, 1, 0, 0, \ldots, 0, 0, 0, 0, 0, 0, \ldots, 0]_k
\end{align*}
\]
Multiple One-hot encodings for one observation

(1) word before; (2) word after

"Corpus"

raw data: sequences of characters

The book was interesting so I was happy.

\[ [0, 0, 0, 0, 1, 0, \ldots, 0]^k \quad [0, \ldots, 0, 1, 0, \ldots, 0]^k \]
Machine Learning: How to setup data

Data

"Corpus"

raw data: sequences of characters

Multiple One-hot encodings for one observation

(1) word before; (2) word after

The book was interesting so I was happy.

\[
\begin{align*}
\text{\textbf{X}} &= [0, 0, 0, 0, 1, 0, \ldots, 0]^k \\
\text{\textbf{Y}} &= [0, \ldots, 0, 1, 0, \ldots, 0]^k \\
\text{\textbf{X}} &\rightarrow \text{\textbf{Y}}
\end{align*}
\]
Machine Learning: How to setup data

Feature Extraction

Multiple One-hot encodings for one observation

(1) word before; (2) word after; (3) percent capitals

“Corpus”

raw data: sequences of characters

The book was *Interesting* so I was happy.

\[
X = \begin{bmatrix}
[0, 0, 0, 0, 1, 0, \ldots, 0]^k \\
[0, 0, 0, 0, 1, 0, \ldots, 0, 0, \ldots, 0, 1, 0, \ldots, 0]^k \\
[0, 0, 0, 0, 1, 0, \ldots, 0, 0, \ldots, 0, 1, 0, \ldots, 0, 0.09]^{2k+1}
\end{bmatrix}
\]

\[
Y = \begin{bmatrix}
\end{bmatrix}
\]
Machine Learning: How to setup data

Data

$X$ $Y$

Model

Does the model hold up?
Machine Learning Goal: Generalize to new data

Training Data

Model

Testing Data

Does the model hold up?

$X \quad Y$
Machine Learning Goal: Generalize to new data

Training Data

Testing Data

80%

20%

$X \quad Y$

Model

Does the model hold up?
Logistic Regression - Regularization

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0</td>
<td>0.6</td>
<td>1</td>
<td>0</td>
<td>0.25</td>
</tr>
<tr>
<td>0</td>
<td>0.5</td>
<td>0.3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0.25</td>
<td>1</td>
<td>1.25</td>
<td>1</td>
<td>0.1</td>
<td>2</td>
</tr>
</tbody>
</table>

\[ X = Y \]
Logistic Regression - Regularization

<table>
<thead>
<tr>
<th>X</th>
<th>=</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0</td>
<td>0.6</td>
</tr>
<tr>
<td>0</td>
<td>0.5</td>
<td>0.3</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.25</td>
<td>1</td>
<td>1.25</td>
</tr>
</tbody>
</table>
Logistic Regression - Regularization

\[
\begin{align*}
X &= Y \\
0.5 & 0 & 0.6 & 1 & 0 & 0.25 \\
0 & 0.5 & 0.3 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0.5 \\
0 & 0 & 0 & 0 & 1 & 1 \\
0.25 & 1 & 1.25 & 1 & 0.1 & 2
\end{align*}
\]

\[
1.2 + -63x_1 + 179x_2 + 71x_3 + 18x_4 + -59x_5 + 19x_6 = \text{logit}(Y)
\]
Logistic Regression - Regularization

\[
X = \begin{bmatrix}
0.5 & 0 & 0.6 & 1 & 0 & 0.25 \\
0 & 0.5 & 0.3 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0.5 \\
0 & 0 & 0 & 0 & 1 & 1 \\
0.25 & 1 & 1.25 & 1 & 0.1 & 2 \\
\end{bmatrix}
\]

\[
\begin{align*}
1.2 + (-63)x_1 + & \quad 179x_2 + 71x_3 + 18x_4 + (-59)x_5 + 19x_6 = \text{logit}(Y) \\
\end{align*}
\]
Logistic Regression - Regularization

\[
1.2 + -63x_1 + 179x_2 + 71x_3 + 18x_4 + -59x_5 + 19x_6 = \operatorname{logit}(Y)
\]

<table>
<thead>
<tr>
<th>(x_1)</th>
<th>(x_2)</th>
<th>(x_3)</th>
<th>(x_4)</th>
<th>(x_5)</th>
<th>(x_6)</th>
<th>(Y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0</td>
<td>0.6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.25</td>
</tr>
<tr>
<td>0</td>
<td>0.5</td>
<td>0.3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0.25</td>
<td>1</td>
<td>1.25</td>
<td>1</td>
<td>0.1</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

"overfitting"
Python Example
Overfitting (1-d non-linear example)

Degree 4
MSE = 4.32e-02(+/- 7.08e-02)
Overfitting (1-d non-linear example)

Degree 1
MSE = 4.08e-01(+/− 4.25e-01)

Degree 4
MSE = 4.32e-02(+/− 7.08e-02)

Underfit

(image credit: Scikit-learn; in practice data are rarely this clear)
Overfitting (1-d non-linear example)

Degree 1
MSE = 4.08e-01(+/− 4.25e-01)

Degree 4
MSE = 4.32e-02(+/− 7.08e-02)

Degree 15
MSE = 1.82e+08(+/− 5.47e+08)

Underfit

Overfit

(image credit: Scikit-learn; in practice data are rarely this clear)
Logistic Regression - Regularization

\[ \begin{align*}
X &= Y \\
\begin{array}{cccccc}
0.5 & 0 & 0.6 & 1 & 0 & 0.25 \\
0 & 0.5 & 0.3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0.5 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0.25 & 1 & 1.25 & 1 & 0.1 & 2 \\
\end{array}
\end{align*} \]

\[ \begin{align*}
1.2 + & -63x_1 + 179x_2 + 71x_3 + 18x_4 + -59x_5 + 19x_6 = \text{logit}(Y)
\end{align*} \]
Logistic Regression - Regularization

What if only 2 predictors?

\[
\begin{align*}
X &= Y \\
\begin{array}{|c|c|c|}
\hline
x_1 & x_2 & \\
0.5 & 0 & 1 \\
0 & 0.5 & 1 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0.25 & 1 & 1 \\
\hline
\end{array}
\end{align*}
\]
Logistic Regression - Regularization

What if only 2 predictors?

A: better fit

\[ 0 + 2x_1 + 2x_2 = \logit(Y) \]
Logistic Regression - Regularization

L1 Regularization - “The Lasso”

*Zeros out* features by adding values that keep from perfectly fitting the data.
Logistic Regression - Regularization

L1 Regularization - “The Lasso”

*Zeros out* features by adding values that keep from perfectly fitting the data.
Logistic Regression - Regularization

L1 Regularization - “The Lasso”
Zeros out features by adding values that keep from perfectly fitting the data.

\[ L(\beta_0, \beta_1, \ldots, \beta_k | X, Y) = \prod_{i=1}^{n} p(x_i)^{y_i} (1 - p(x_i))^{1-y_i} \]

set betas that maximize \( L \)
Logistic Regression - Regularization

L1 Regularization - “The Lasso”

Zeros out features by adding values that keep from perfectly fitting the data.

\[
L(\beta_0, \beta_1, \ldots, \beta_k \mid X, Y) = \prod_{i=1}^{n} p(x_i)^{y_i} (1 - p(x_i))^{1-y_i} - \frac{1}{C} \sum_{j=1}^{m} |\beta_j|
\]

set betas that maximize penalized \( L \)
Logistic Regression - Regularization

**L1 Regularization - “The Lasso”**

Zeros out features by adding values that keep from perfectly fitting the data.

$$L(\beta_0, \beta_1, \ldots, \beta_k \mid X, Y) = \prod_{i=1}^{n} p(x_i)^{y_i} (1 - p(x_i))^{1-y_i} - \frac{1}{C} \sum_{j=1}^{m} |\beta_j|$$

set betas that maximize **penalized L**

Sometimes written as:

$$||\beta||_1$$
Logistic Regression - Regularization

L2 Regularization - “Ridge”

*Shrinks* features by adding values that keep from perfectly fitting the data.

\[
L(\beta_0, \beta_1, \ldots, \beta_k | X, Y) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i} - \frac{1}{C} \sum_{j=1}^{m} \beta_j^2
\]

set betas that maximize *penalized* L
Machine Learning Goal: Generalize to new data

Training Data

Testing Data

Does the model hold up?
Machine Learning Goal: Generalize to new data

Training Data

Development Set

Testing Data

Set penalty

Model

Does the model hold up?
Logistic Regression - Review

- Classification: \( P(Y \mid X) \)
- Learn logistic curve based on example data
  - training + development + testing data
- Set betas based on maximizing the likelihood
  - "shifts" and "twists" the logistic curve
- Multivariate features: One-hot encodings
- Separation represented by hyperplane
- Overfitting
- Regularization
Example

See [notebook](http://example.com) on website.

In [44]: %matplotlib inline
   
   # above allows plots to display on the screen.
   
   # python includes
   import sys
   
   # standard probability includes:
   import numpy as np  # matrices and data structures
   import scipy.stats as ss  # standard statistical operations
   import pandas as pd  # keeps data organized, works well with data
   import matplotlib
   import matplotlib.pyplot as plt  # plot visualization

In [53]: # let's just look at what happens to the logit function as we change the beta coefficients
   
   def logistic_function(x):
       return np.exp(x) / (1+np.exp(x))
   
   def logistic_function_with_betas(x, beta0=0, beta1=1):
       # now using linear function: beta0 + beta1*x for the exponent:
       return np.exp(beta0 + beta1*x) / (1+np.exp(beta0 + beta1*x))

   xpoints = np.linspace(-10, 10, 100)
   plt.plot(xpoints, [logistic_function(x) for x in xpoints])
   plt.plot(xpoints, [logistic_function_with_betas(x, 2, 1) for x in xpoints])  # shifts the intercept with zero
   plt.plot(xpoints, [logistic_function_with_betas(x, 0, 3.14591459653) for x in xpoints])  # twists the line vertically
   plt.plot(xpoints, [logistic_function_with_betas(x, 0, 1/3.14591459653) for x in xpoints])  # twists it horizontally

Out[53]: [mpl_toolkits.axisartist.axislines.Line2D at 0x2591f435f60]
Extra Material

One approach to finding the parameters which maximize the likelihood function...
"best fit" : whatever maximizes the likelihood function:

\[
L(\beta_0, \beta_1, \ldots, \beta_k | X, Y) = \prod_{i=1}^{n} p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}
\]

\[
p_i \equiv P(Y_i = 1 | X_i = x) = \frac{e^{\beta_0+\beta_1x_i}}{1 + e^{\beta_0+\beta_1x_i}}
\]

To estimate \( \beta \), one can use **reweighted least squares**:

1. Calculate \( p_i \) and let \( W \) be a diagonal matrix
   where element \((i, i) = p_i(1 - p_i)\).
2. Set \( z_i = \text{logit}(p_i) + \frac{Y_i - p_i}{p_i(1 - p_i)} = X\hat{\beta} + \frac{Y_i - p_i}{p_i(1 - p_i)} \)
3. Set \( \hat{\beta} = (X^TWX)^{-1}X^TWz \) //weighted lin. reg. of \( Z \) on \( Y \).
4. Repeat from 1 until \( \hat{\beta} \) converges.

(Wasserman, 2005; Li, 2010)
“best fit” : whatever maximizes the likelihood function:

\[
L(\beta_0, \beta_1, \ldots, \beta_k | X, Y) = \prod_{i=1}^{n} p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}
\]

This is just one way of finding the betas that maximize the likelihood function. In practice, we will use existing libraries that are fast and support additional useful steps like **regularization**.

To estimate \( \beta \), one can use **reweighted least squares**:

1. Calculate \( p_i \) and let \( W \) be a diagonal matrix where element \( (i, i) = p_i(1 - p_i) \).
2. Set \( z_i = \text{logit}(p_i) + \frac{Y_i - p_i}{p_i(1 - p_i)} = X\hat{\beta} + \frac{Y_i - p_i}{p_i(1 - p_i)} \)
3. Set \( \hat{\beta} = (X^T W X)^{-1} X^T W z / \text{weighted lin. reg. of } Z \text{ on } Y \).
4. Repeat from 1 until \( \hat{\beta} \) converges.

(Wasserman, 2005; Li, 2010)