Text Classification: Lexicon-Based and Supervised Logistic Regression

CSE354 - Spring 2021
NLP’s practical applications

● Machine translation
● Automatic speech recognition
  ○ Personalized assistants
  ○ Auto customer service
● Information Retrieval
  ○ Web Search
  ○ Question Answering
● Text Categorization:
  e.g. Sentiment Analysis
● Computational Social Science

how?

● 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
- Text Categorization: e.g. Sentiment Analysis
- Computational Social Science

how?

- Lexica
- 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

- **Lexicon-based classification (closed-vocabulary)**
- **Supervised classification (open-vocabulary)**
  - Goal of logistic regression
  - The “loss function” -- what logistic regression tries to optimize
  - Logistic regression with multiple features
  - How to evaluation: Training and test datasets
  - Overfitting: role of regularization
The Buccaneers win it!

President Biden vetoed bill

Twitter to be acquired by Apple

I like the movie. The movie is like terrible.
Sentiment

“I bought an iPhone a few days ago. It was such a nice phone. The touch screen was really cool. The voice quality was clear too. Although the battery life was not long, that is ok for me. ...” (Liu, 2010)
"I bought an iPhone a few days ago. It was such a nice phone. The touch screen was really cool. The voice quality was clear too. Although the battery life was not long, that is ok for me. ..."

"This past Saturday, I bought a Nokia phone and my girlfriend bought a Motorola phone. We called each other when we got home. The voice on my phone was not so clear, worse than my previous phone. ..." (Liu, 2010)
Lexica


Table 1: Paradigm word list.

<table>
<thead>
<tr>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>dazzling, brilliant, phenomenal, excellent</td>
<td>sucks, terrible, awful, unwatchable, hideous</td>
</tr>
<tr>
<td>fantastic, gripping, mesmerizing, riveting</td>
<td>hideous, bad, cliched, boring, stupid, slow</td>
</tr>
<tr>
<td>spectacular, cool</td>
<td>worst, waste, unexcit, rubbish, tedious</td>
</tr>
<tr>
<td>exciting, love, wonderful, best, great</td>
<td>unbearable, pointless, cheesy, frustrated</td>
</tr>
<tr>
<td>still, beautiful</td>
<td>awkward, disappointing</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Proposed word lists</th>
<th>Accuracy</th>
<th>Ties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human 1 positive: dazzling, brilliant, phenomenal, excellent, fantastic</td>
<td>58%</td>
<td>75%</td>
</tr>
<tr>
<td>negative: suck, terrible, awful, unwatchable, hideous</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Human 2 positive: gripping, mesmerizing, riveting, spectacular, cool, awesome, thrilling, badass, excellent, moving, exciting</td>
<td>64%</td>
<td>39%</td>
</tr>
<tr>
<td>negative: bad, cliched, sucks, boring, stupid, slow</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: Baseline results for human word lists. Data: 700 positive and 700 negative reviews.

Sentiment Analysis

“I bought an iPhone a few days ago. It was such a nice phone. The touch screen was really cool. The voice quality was clear too. Although the battery life was not long, that is ok for me. ..." (Liu, 2010)
Sentiment Analysis

“I bought an iPhone a few days ago. It was such a nice phone. The touch screen was really cool. The voice quality was clear too. Although the battery life was not long, that is ok for me. ...” (Liu, 2010)

positive lexicon
{'like', 'nice', 'good', 'cool', 'quality', 'excellent', ...}

negative lexicon
{'dislike', 'bad', 'not', 'horrible', 'terrible', 'awful', 'ok', ...}
"I bought an iPhone a few days ago. It was such a nice phone. The touch screen was really cool. The voice quality was clear too. Although the battery life was not long, that is ok for me. ..." (Liu, 2010)

**Sentiment Analysis**

```
positive lexicon
{'like', 'nice', 'good', 'cool', 'quality', 'excellent', ...}

negative lexicon
{'dislike', 'bad', 'not', 'horrible', 'terrible', 'awful', 'ok', ...}
```

```
words = tokenize(message)
pos_count = len([word for word in words if word in pos_set])
neg_count = len([word for word in words if word in neg_set])
score = pos_count - neg_count
```
Sentiment Analysis

“I bought an iPhone a few days ago. It was such a nice phone. The touch screen was really cool. The voice quality was clear too. Although the battery life was not long, that is ok for me. ...” (Liu, 2010)

words = tokenize(message)

pos_count = len([word for word in words if word in pos_set])
neg_count = len([word for word in words if word in neg_set])

pos_p = pos_count/len(words)

neg_p = neg_count/len(words)

score = pos_p - neg_p

if (score > 0): return "positive"
else: return "negative"
"I bought an iPhone a few days ago. It was such a nice phone. The touch screen was really cool. The voice quality was clear too. Although the battery life was not long, that is ok for me. ..." (Liu, 2010)

Sentiment Analysis

- **positive lexicon**:
  - {'like', 'nice', 'good', 'cool', 'quality', 'excellent', ...}

- **negative lexicon**:
  - {'dislike', 'bad', 'not', 'horrible', 'terrible', 'awful', 'ok', ...}

Classifier:

```python
words = tokenize(message)
pos_count = len([word for word in words if word in pos_set])
neg_count = len([word for word in words if word in neg_set])
pos_p = pos_count / len(words)
neg_p = neg_count / len(words)

score = pos_p - neg_p
if (score > thresh):
    return "positive"
else:
    return "negative"
```
Sentiment Analysis

“This past Saturday, I bought a Nokia phone and my girlfriend bought a Motorola phone. We called each other when we got home. The voice on my phone was not so clear, worse than my previous phone. …"

```python
words = tokenize(message)
pos_count = len([word for word in words if word in pos_set])
neg_count = len([word for word in words if word in neg_set])
pos_p = pos_count/len(words)
neg_p = neg_count/len(words)

score = pos_p - neg_count
if (score > thresh): return "positive"
else: return "negative"
```
Sentiment Analysis

“I bought an iPhone a few days ago. It was such a nice phone. The touch screen was really cool. The voice quality was clear too. Although the battery life was not long, that is ok for me. ...” (Liu, 2010)
Sentiment Analysis

“I bought an iPhone a few days ago. It was such a nice phone. The touch screen was really cool. The voice quality was clear too. Although the battery life was not long, that is ok for me. However, my mother was mad with me as I did not tell her before I bought it. She also thought the phone was too expensive, and wanted me to return it to the shop." (Liu, 2010)
Sentiment Analysis

“This past Saturday, I bought a Nokia phone and my girlfriend bought a Motorola phone. We called each other when we got home. The voice on my phone was not so clear, worse than my previous phone. The camera was good. My girlfriend was quite happy with her phone. I wanted a phone with good voice quality. So my purchase was a real disappointment. I returned the phone yesterday.”(Liu, 2010)
Sentiment -- Using Statistics

<table>
<thead>
<tr>
<th>Proposed word lists</th>
<th>Accuracy</th>
<th>Ties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human 1</td>
<td>58%</td>
<td>75%</td>
</tr>
<tr>
<td>positive: dazzling, brilliant, phenomenal, excellent, fantastic negative: suck, terrible, awful, unwatchable, hideous</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Human 2</td>
<td>64%</td>
<td>39%</td>
</tr>
<tr>
<td>positive: gripping, mesmerizing, riveting, spectacular, cool, awesome, thrilling, badass, excellent, moving, exciting negative: bad, cliched, sucks, boring, stupid, slow</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: Baseline results for human word lists. Data: 700 positive and 700 negative reviews.

<table>
<thead>
<tr>
<th>Proposed word lists</th>
<th>Accuracy</th>
<th>Ties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human 3 + stats</td>
<td>69%</td>
<td>16%</td>
</tr>
<tr>
<td>positive: love, wonderful, best, great, superb, still, beautiful negative: bad, worst, stupid, waste, boring, ?, !</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2: Results for baseline using introspection and simple statistics of the data (including test data).

Sentiment -- Using Statistics

<table>
<thead>
<tr>
<th>Proposed word lists</th>
<th>Accuracy</th>
<th>Ties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Human 1</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>positive: dazzling, brilliant, phenomenal, excellent, fantastic</td>
<td>58%</td>
<td>75%</td>
</tr>
<tr>
<td>negative: suck, terrible, awful, unwatchable, hideous</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Human 2</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>positive: gripping, mesmerizing, riveting, spectacular, cool, awesome, thrilling, badass, excellent, moving, exciting</td>
<td>64%</td>
<td>39%</td>
</tr>
<tr>
<td>negative: bad, cliched, sucks, boring, stupid, slow</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: Baseline results for human word lists. Data: 700 positive and 700 negative reviews.

<table>
<thead>
<tr>
<th>Proposed word lists</th>
<th>Accuracy</th>
<th>Ties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Human 3 + stats</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>positive: love, wonderful, best, great, superb, still, beautiful</td>
<td>69%</td>
<td>16%</td>
</tr>
<tr>
<td>negative: bad, worst, stupid, waste, boring, 😞🤔🙁</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2: Results for baseline using introspection and simple statistics of the data (including test data).

Figure 2: Results for baseline using introspection and simple statistics of the data (including test data).
Supervised Classification
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 \( x_i \).

\[
f(X) = Y
\]

\[
\begin{array}{c|c|c}
 i & X & Y \\
 0 & 0.0 & 0 \\
 1 & 0.5 & 0 \\
 2 & 1.0 & 1 \\
 3 & 0.25 & 0 \\
 4 & 0.75 & 1 \\
\end{array}
\]
Supervised Classification

\( X \) - features of \( N \) observations (e.g., 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>

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. (The alternative, *unsupervised* machine learning, tries to learn with only an \(X\)).

*Classification*: The outcome (\(Y\)) is a discrete class.

For example: \(y \in \{\text{noun, verb, adjective, adverb}\}\)

\(y \in \{\text{positive\_sentiment, negative\_sentiment}\}\).
Classification as Producing a Probability

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
Classification as Producing a Probability

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.
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.
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:
\[
\begin{align*}
Y & : 1 \text{ if target is verb, 0 otherwise;} \\
X & : 1 \text{ if “was” occurs before target; 0 otherwise}
\end{align*}
\]

\[I \text{ was } \textit{reading} \text{ for NLP.} \quad \text{We were } \textit{fine}. \quad \text{I am } \textit{good}.\]

\[\text{The cat was } \textit{very} \text{ happy.} \quad \text{We enjoyed the } \textit{reading} \text{ material.} \quad \text{I was } \textit{good}.\]
Classification as Producing a Probability

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.
Classification as Producing a Probability

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.*
Classification as Producing a Probability

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.
Classification as Producing a Probability

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

<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.

<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.

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

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.

<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.

<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>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.

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.

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.

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.

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

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

In [81]: 1 -b2_0/b2_1
Out[81]: 0.3108930938058134

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

optimal $b_0, b_1$ changed!
Logistic Regression on a single feature \((x)\)

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

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

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

\[
P(Y_i = 1 | X_i = x) = \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}} = \frac{1}{1 + e^{-(\beta_0 + \sum_{j=1}^{m} \beta_j x_{ij})}}
\]
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.

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

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: $X_i, B_0, B_1$.
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: \(X_i, B_0, B_1\)

\(X\) 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.

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

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.
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: $X$, $B_0$, and $B_1$.

$X$ is given. $B_0$ and $B_1$ 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 $Y$).
"best fit" : whatever maximizes the likelihood function:

$$L(\beta_0, \beta_1 | X, Y) = \prod_{i=1}^{n} p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}$$
"best fit": whatever maximizes the likelihood function:

\[
L(\beta_0, \beta_1 | X, Y) = \prod_{i=1}^{n} p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}
\]
“best fit” : whatever maximizes the likelihood function:

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

“best fit” : more efficient to maximize log likelihood :
Logistic Regression on a single feature $(x)$

"best fit": whatever maximizes the *likelihood* function:

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

"best fit": more efficient to maximize *log likelihood*:

$$\ell(\beta) = \sum_{i=1}^{N} y_i \log p(x_i) + (1-y_i) \log (1-p(x_i))$$
“best fit”: whatever maximizes the likelihood function:

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

“best fit”: more efficient to maximize log likelihood:

$$\ell(\beta) = \sum_{i=1}^{N} y_i \log p(x_i) + (1-y_i) \log (1-p(x_i))$$

“best fit” for neural networks: software designed to **minimize** rather than maximize (typically, normalized by N, the number of examples.)
“best fit”: whatever maximizes the likelihood function:

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

“best fit”: more efficient to maximize log likelihood:

\[
\ell(\beta) = \sum_{i=1}^{N} y_i \log p(x_i) + (1-y_i) \log (1-p(x_i))
\]

“best fit” for neural networks: software designed to minimize rather than maximize (typically, normalized by N, number of examples.) "log loss" or "normalized log loss":

\[
J(\beta) = -\frac{1}{N} \sum_{i=1}^{N} y_i \log p(x_i) + (1 - y_i) \log (1 - p(x_i))
\]
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\}
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

\[ Y_i \in \{0, 1\}; \ X \text{ can be anything numeric.} \]

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

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

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

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

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

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.
Logistic Regression

Example:  
Y: 1 if target is a part of a proper noun, 0 otherwise;  
X1: number of capital letters in target and surrounding words.  
X2: does the target word start with a capital letter?

Let's add a feature!

<table>
<thead>
<tr>
<th>x2</th>
<th>x1</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
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>

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>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>1</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>0.35</td>
<td>1</td>
</tr>
</tbody>
</table>
```

“Corpus”

raw data: sequences of characters
Machine Learning: How to setup data

Feature Extraction
--pull out observations and feature vector per observation.

```
<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>
```

“Corpus”
raw data: sequences of characters
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>
```

"Corpus"

raw data: sequences of characters
Machine Learning: How to setup data

Feature Extraction
--pull out observations and feature vector per observation.

<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>1</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
- *k* features indicating whether *k* words were mentioned or not

<table>
<thead>
<tr>
<th></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>1</td>
</tr>
<tr>
<td>N</td>
<td>0.35</td>
<td>0</td>
</tr>
</tbody>
</table>

training
Machine Learning: How to setup data

Feature Extraction

Data

Multi-hot Encoding
- Each word gets an index in the vector
- "Corpus" raw data: sequences of characters
- 1 if present; 0 if not
- Number of features; e.g.
  - Number of capital letters
  - Whether "I" was mentioned or not
  - $k$ features indicating whether $k$ words were mentioned or not

$X$ $Y$
Machine Learning: How to setup data

Data

Feature Extraction

Corpus
raw data: sequences of characters

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.

The book was interesting so I was happy.

Data

- 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
- 1 if present; 0 if not

**Feature example: is word present in document?**

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

raw data: sequences of characters

```
[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
- “Corpus” raw data: sequences of characters

Feature example: is word present in document

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

\[
\text{sad}
\]

The book was interesting so I was happy.
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

Feature example: is previous word “the”?

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

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

\( k \) 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
- “Curious” → 1 if present; 0 if not
- Feature example: is previous word “the”?

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

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

\( k \) 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”?  

raw data: sequences of characters

Data

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

\(k\) 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
- All indices 0 except present word

Feature example: which is previous word?

raw data: sequences of characters

Data

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
</tr>
</thead>
</table>

The book was interesting so I was happy.

Feature example: which is previous word?

$$[0, 1, 0, 0, 0, \ldots, 0, 0, 0, 0, 0, 0, 0, 0, \ldots, 0]^k$$

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

$[0, 1, 0, 0, 0, \ldots, 0, 0, 0, 0, 0, 0, \ldots, 0]^k$

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

**Feature Extraction**

**Multiple One-hot encodings for one observation**

(1) word before; (2) word after

"Corpus"

raw data: sequences of characters

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

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

*The book was interesting so I was happy.*
Machine Learning: How to setup data

**Feature Extraction**

**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.

$$[0, 0, 0, 0, 1, 0, \ldots, 0]^k \ [0, \ldots, 0, 1, 0, \ldots, 0]^k = \ [0, 0, 0, 0, 1, 0, \ldots, 0, \ldots, 0, 1, 0, \ldots, 0]^{2k}$$
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.

$$[0, 0, 0, 0, 1, 0, \ldots, 0]^k [0, \ldots, 0, 1, 0, \ldots, 0]^k =$$

$$[0, 0, 0, 0, 1, 0, \ldots, 0, 0, \ldots, 0, 1, 0, \ldots, 0]^{2k}$$

$$[0, 0, 0, 0, 1, 0, \ldots, 0, 0, \ldots, 0, 1, 0, \ldots, 0, 0.09]^{2k+1}$$
Machine Learning: How to setup data

$X$ $Y$

Data

Model

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

Training Data

Model

Testing Data

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

80% Training Data

20% Testing Data

Does the model hold up?

$X$ $Y$
## Logistic Regression - Regularization

$$X \begin{array}{cccccc} 0.5 & 0 & 0.6 & 1 & 0 & 0.25 \\ 0 & 0.5 & 0.3 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0.5 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0.25 & 1 & 1.25 & 1 & 0.1 & 2 \\ \end{array} = Y \begin{array}{c} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ \end{array}$$
Logistic Regression - Regularization

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></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>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0.5</td>
<td>0.3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.5</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</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>
Logistic Regression - Regularization

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

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

\[\begin{array}{cccccc}
 x_1 & x_2 & \ldots & X & = & Y \\
 0.5 & 0 & 0.6 & 1 & 0 & 0.25 \\
 0 & 0.5 & 0.3 & 0 & 0 & 0 \\
 0 & 0 & 1 & 1 & 1 & 0.5 \\
 0 & 0 & 0 & 0 & 1 & 1 \\
 0.25 & 1 & 1.25 & 1 & 0.1 & 2
\end{array}\]

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

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

\[ \begin{align*}
Y &= \begin{bmatrix}
1 \\
1 \\
0 \\
0 \\
1 \\
\end{bmatrix}
\end{align*} \]

Row 2 is marked with "overfitting".

\[ 1.2 + -63x_1 + \boxed{179x_2} + 71x_3 + 18x_4 + -59x_5 + 19x_6 = \text{logit}(Y) \]
Python Example
Overfitting (1-d non-linear example)
Overfitting (1-d non-linear example)

Underfit

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

Underfit

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)

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

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

"overfitting"

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

“overfitting”: generally due to trying to fit too many features given the number of observations.

\[
1.2 + (-63)x_1 + 17.1x_2 + 71x_3 + 18x_4 + (-59)x_5 + 19x_6 = \text{logit}(Y)
\]
Logistic Regression - Regularization

What if only 2 predictors?

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.25</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Logistic Regression - Regularization

What if only 2 predictors?

A: better fit
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.

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

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.

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

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

L2 Regularization - “Ridge”

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

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

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

Training Data

Model

Does the model hold up?

Testing Data

X

Y

80%

20%
Machine Learning Goal: Generalize to new data

Training Data

Development

Testing Data

Set penalty

Model

Does the model hold up?
Example

See notebook on website.

```python
# Let's just look at what happens to the logit function as we change the beta coefficients

In [53]:
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]: [matplotlib.lines.Line2D at 0x2691f435f60>]

For 2021: add multinomial
Logistic Regression - Review

- Probabilistic Classification: \( P(Y \mid X) \)
- Learn logistic curve based on example data
  - training + development + testing data
- Set betas based on maximizing the likelihood
  - or based on minimizing log loss
    - “shifts” and “twists” the logistic curve
    - separation represented by hyperplane at 0.50
- Multivariate features: One-hot encodings
- Overfitting and Regularization
Extra Material
The goal of this function is to:
take in the variable $x$
return a probability that $Y$ is 1.

Note that there are only three variables on the right:
$X$, $B_0$, $B_1$
$X$ is given.
$B_0$ and $B_1$ must be learned.

Logistic Regression on a single feature ($x$)

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

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 $z_i = \logit(p_i) + \frac{Y_i - p_i}{p_i(1 - p_i)} = X\hat{\beta} + \frac{Y_i - p_i}{p_i(1 - p_i)}$
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)
The goal of this function is to: take in the variable $x$ and return a probability that $Y$ is 1.

Alternative to gradient descent:

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

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 $\hat{\beta} = (X^T W X)^{-1} X^T W z$ //weighted lin. reg. of $Z$ on $Y$.
4. Repeat from 1 until $\hat{\beta}$ converges.

(Wasserman, 2005; Li, 2010)