In our example, we want to classify a restaurant review as positive or negative.

Sentence from review → Classifie r Model → Predicted class

Input: x

“sentence”

Features
Awesome

Output: y

Predicted class
Consider if only two words had non-zero coefficients

\[
\hat{s} = 1 \cdot \text{#awesome} - 1.5 \cdot \text{#awful}
\]

<table>
<thead>
<tr>
<th>Word</th>
<th>Coefficient</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(w_0)</td>
<td>0.0</td>
</tr>
<tr>
<td>awesome</td>
<td>(w_1)</td>
<td>1.0</td>
</tr>
<tr>
<td>awful</td>
<td>(w_2)</td>
<td>-1.5</td>
</tr>
</tbody>
</table>
One idea is to just model the processing of finding $\hat{w}$ based on what we discussed in linear regression:

$$\hat{w} = \arg\min_w \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\{y_i \neq \hat{y}_i\}$$

Will this work?

Assume $h_1(x) = \#awesome$ so $w_1$ is its coefficient and $w_2$ is fixed.

![Graph showing #awful vs. error and #awesome vs. w1](image-url)
Minimizing classification error is probably the most intuitive thing to do given all we have learned from regression. However, it just doesn’t work in this case with classification.

We aren’t able to use a method like gradient descent here because the function isn’t “nice” (it’s not continuous, it’s not differentiable, etc.).

We will use a stand-in for classification error that will allow us to use an optimization algorithm. But first, we have to change the problem we care about a bit.

Instead of caring about the classifications, let’s look at some probabilities
Probabilities

Assume that there is some randomness in the world, and instead will try to model the probability of a positive/negative label.

Examples:

“The sushi & everything else were awesome!”

Definite positive (+1)

\[
P(y = +1 \mid x = \text{“The sushi & everything else were awesome!”}) = 0.99
\]

“The sushi was alright, the service was OK”

Not as sure

\[
P(y = -1 \mid x = \text{“The sushi alright, the service was okay!”}) = 0.5
\]

Use probability as the measurement of certainty

\[P(y \mid x)\]
Idea: Estimate probabilities $\hat{P}(y|x)$ and use those for prediction.

Probability Classifier

Input $x$: Sentence from review

- Estimate class probability $\hat{P}(y = +1|x)$

  - If $\hat{P}(y = +1|x) > 0.5$:
    - $\hat{y} = +1$
  
  - Else:
    - $\hat{y} = -1$

Notes:

- Estimating the probability improves **interpretability**
Idea: Let’s try to relate the value of $\text{Score}(x)$ to $\hat{P}(y = +1|x)$

What if $\text{Score}(x)$ is positive?

$P(y = +1|x) > \frac{1}{2}$

What if $\text{Score}(x)$ is negative?

$P(y = -1|x) > \frac{1}{2}$

$P(y = +1|x) < \frac{1}{2}$

What if $\text{Score}(x)$ is 0?

$P(y = +1|x) = \frac{1}{2}$
Interpreting Score

\[ \text{Score}(x_i) = w^T h(x_i) \]

\[ \hat{y}_i = -1 \]

\[ \hat{y}_i = +1 \]

**Very sure**

\[ \hat{y}_i = -1 \]

\[ \hat{y}_i = +1 \]

**Not sure if**

\[ \hat{y}_i = -1 \text{ or } \hat{y}_i = +1 \]

\[ \hat{P}(y_i = +1 | x_i) = 0 \]

\[ \hat{P}(y_i = +1 | x_i) = 0.5 \]

\[ \hat{P}(y_i = +1 | x_i) = 1 \]

\[ \hat{P}(y = +1 | x) \]

**Very sure**

\[ \hat{y}_i = +1 \]

\[ \hat{P}(y_i = +1 | x_i) = 1 \]
Logistic Function

Use a function that takes numbers arbitrarily large/small and maps them between 0 and 1.

\[ \text{sigmoid}(\text{Score}(x)) = \frac{1}{1 + e^{-\text{Score}(x)}} \]

### Table: \( \text{sigmoid}(\text{Score}(x)) \)

<table>
<thead>
<tr>
<th>Score(x)</th>
<th>sigmoid(Score(x))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-\infty)</td>
<td>( \frac{1}{1 + e^\infty} = \frac{1}{1 + \infty} = 0 )</td>
</tr>
<tr>
<td>(-2)</td>
<td>( \approx 0.12 )</td>
</tr>
<tr>
<td>0</td>
<td>( \frac{1}{1 + e^0} = \frac{1}{1 + 1} = \frac{1}{2} )</td>
</tr>
<tr>
<td>2</td>
<td>( \approx 0.98 )</td>
</tr>
<tr>
<td>( \infty )</td>
<td>( \frac{1}{1 + e^{-\infty}} = \frac{1}{1 + \frac{1}{e^{-\infty}}} = \frac{1}{1 + 1} = \frac{1}{2} )</td>
</tr>
</tbody>
</table>
Logistic Regression Model

Logistic Regression Classifier

Input $x$: Sentence from review

Estimate class probability $\hat{P}(y = +1|x, \hat{w}) = sigmoid(\hat{w}^T h(x))$

If $\hat{P}(y = +1|x, \hat{w}) > 0.5$:
- $\hat{y} = +1$

Else:
- $\hat{y} = -1$

\[ Score(x) = w^T h(x) \]

\[ P(y_i = +1|x_i, w) = sigmoid(Score(x_i)) = \frac{1}{1 + e^{-w^T h(x_i)}} \]
\[ \hat{P}(y = +1|x, \hat{w}) = \text{sigmoid}(\hat{w}^T h(x)) = \frac{1}{1 + e^{-\hat{w}^T h(x)}} \]
CSE/STAT 416

Logistic Regression

Tanmay Shah
University of Washington
June 3, 2024

❓ Questions? Raise hand or sli.do #cs416
镫 Before Class: Does a straw have two holes or one?
🎵 Listening to: New Jeans
Coming up
- Week 3: Societal Impacts of ML (Fairness and Bias)
- Week 4: Other ML models for classification
- Week 5: Deep Learning

HW3 released today, due next Tuesday

Midterm
- Released Friday 4/21 at 8:30 am. Due Monday 4/24 at 11:59 pm.
  - Untimed, but would be good to time yourself as practice for the final
  - Should take ~1 hour if you know the material
- Format: Think longer conceptual assignment from HW
- Covers everything from Module 0 (Regression) to Module 3 (Societal Impact, Bias, Fairness)
- Should follow our normal collaboration policy
  - Think of it as a trial run for the final exam
For binary classification, there are only two types of mistakes:

\[ \hat{y} = +1, \quad y = -1 \]
\[ \hat{y} = -1, \quad y = +1 \]

Generally we make a **confusion matrix** to understand mistakes.

Tip on remembering: complete the sentence “My prediction was a ...”
Confusion Matrix Example

Predicted Label

True Positive (TP)  False Negative (FN)

False Positive (FP)  True Negative (TN)
Which is Worse?

What’s worse, a false negative or a false positive?
It entirely depends on your application!

Detecting Spam
False Negative: Annoying
False Positive: Email lost

Medical Diagnosis
False Negative: Disease not treated
False Positive: Wasteful treatment

In almost every case, how treat errors depends on your context.
We mentioned on the first day how ML is being used in many contexts that impact crucial aspects of our lives.

Models making errors is a given, what we do about that is a choice:

- Are the errors consequential enough that we shouldn’t use a model in the first place?
- Do different demographic groups experience errors at different rates?
  - If so, we would hopefully want to avoid that model!

Will talk more about how to define whether or not a model is fair / discriminatory next week. Will use these notions of error as a starting point!
Binary Classification Measures

Notation

\[ C_{TP} = \#TP, \quad C_{FP} = \#FP, \quad C_{TN} = \#TN, \quad C_{FN} = \#FN \]
\[ N = C_{TP} + C_{FP} + C_{TN} + C_{FN} \]
\[ N_P = C_{TP} + C_{FN}, \quad N_N = C_{FP} + C_{TN} \]

Error Rate

\[ \frac{C_{FP} + C_{FN}}{N} \]

Accuracy Rate

\[ \frac{C_{TP} + C_{TN}}{N} \]

False Positive Rate (FPR)

\[ \frac{C_{FP}}{N_N} \]

False Negative Rate (FNR)

\[ \frac{C_{FN}}{N_P} \]

True Positive Rate or Recall

\[ \frac{C_{TP}}{N_P} \]

Precision

\[ \frac{C_{TP}}{C_{TP} + C_{FP}} \]

F1-Score

\[ \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \]

See more!
Consider predicting *(Healthy, Cold, Flu)*

<table>
<thead>
<tr>
<th>True Label</th>
<th>Predicted Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Healthy</td>
<td>Healthy</td>
</tr>
<tr>
<td>Healthy</td>
<td>60</td>
</tr>
<tr>
<td>Cold</td>
<td>4</td>
</tr>
<tr>
<td>Flu</td>
<td>0</td>
</tr>
</tbody>
</table>
Suppose we trained a classifier and computed its confusion matrix on the training dataset. **Is there a class imbalance in the dataset and if so, which class has the highest representation?**

<table>
<thead>
<tr>
<th>Predicted Label</th>
<th>Pupper</th>
<th>Doggo</th>
<th>Woofer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pupper</td>
<td>2</td>
<td>27</td>
<td>4</td>
</tr>
<tr>
<td>Doggo</td>
<td>4</td>
<td>25</td>
<td>4</td>
</tr>
<tr>
<td>Woofer</td>
<td>1</td>
<td>30</td>
<td>2</td>
</tr>
</tbody>
</table>
Suppose we trained a classifier and computed its confusion matrix on the training dataset. **Is there a class imbalance in the dataset and if so, which class has the highest representation?**

<table>
<thead>
<tr>
<th>True Label</th>
<th>Predicted Label</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pupper</td>
</tr>
<tr>
<td>Pupper</td>
<td>2</td>
</tr>
<tr>
<td>Doggo</td>
<td>4</td>
</tr>
<tr>
<td>Woofer</td>
<td>1</td>
</tr>
</tbody>
</table>
How much data?

The more the merrier
    But data quality is also an extremely important factor

Theoretical techniques can bound how much data is needed
    Typically too loose for practical applications
    But does provide some theoretical guarantee

In practice
    More complex models need more data
Learning Curve

How does the true error of a model relate to the amount of training data we give it?

Hint: We’ve seen this picture before
Learning Curve

What if we use a more complex model?
Theme: Describe high level idea and metrics for classification

Ideas:

- Applications of classification
- Linear classifier
- Decision boundaries
- Classification error / Classification accuracy
- Class imbalance
- Confusion matrix
- Learning theory
One idea is to just model the processing of finding $\hat{w}$ based on what we discussed in linear regression:

$$\hat{w} = \arg\min_w \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\{y_i \neq \hat{y}_i\}$$

Will this work?

Assume $h_1(x) = \#awesome$ so $w_1$ is its coefficient and $w_2$ is fixed.
Logistic Regression Model

Logistic Regression Classifier

Input $x$: Sentence from review

Estimate class probability $\hat{P}(y = +1 | x, \hat{w}) = sigmoid(\hat{w}^T h(x))$

If $\hat{P}(y = +1 | x, \hat{w}) > 0.5$: 
- $\hat{y} = +1$

Else:
- $\hat{y} = -1$

$$P(y_i = +1 | x_i, w) = sigmoid(Score(x_i)) = \frac{1}{1 + e^{-w^T h(x_i)}}$$
Demo

Show logistic demo (see course website)
Think

What would the Logistic Regression model predict for $P(y = -1 | x, w)$?

"Sushi was great, the food was awesome, but the service was terrible"

<table>
<thead>
<tr>
<th>$h_1(x)$</th>
<th>$h_2(x)$</th>
<th>$h_3(x)$</th>
<th>$h_4(x)$</th>
<th>$h_5(x)$</th>
<th>$h_6(x)$</th>
<th>$h_7(x)$</th>
<th>$h_8(x)$</th>
<th>$h_9(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>sushi</td>
<td>was</td>
<td>great</td>
<td>the</td>
<td>food</td>
<td>awesome</td>
<td>but</td>
<td>service</td>
<td>terrible</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Word | Weight
--- | ---
{sushi} | 0
{was} | 0
{great} | 1
{the} | 0
{food} | 0
{awesome} | 2
{but} | 0
{service} | 0
{terrible} | −1
What would the Logistic Regression model predict for $P(y = -1 \mid x, w)$?

"Sushi was great, the food was awesome, but the service was terrible”

<table>
<thead>
<tr>
<th>Word</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>sushi</td>
<td>0</td>
</tr>
<tr>
<td>was</td>
<td>0</td>
</tr>
<tr>
<td>great</td>
<td>1</td>
</tr>
<tr>
<td>the</td>
<td>0</td>
</tr>
<tr>
<td>food</td>
<td>0</td>
</tr>
<tr>
<td>awesome</td>
<td>2</td>
</tr>
<tr>
<td>but</td>
<td>0</td>
</tr>
<tr>
<td>service</td>
<td>0</td>
</tr>
<tr>
<td>terrible</td>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$h_1(x)$</th>
<th>$h_2(x)$</th>
<th>$h_3(x)$</th>
<th>$h_4(x)$</th>
<th>$h_5(x)$</th>
<th>$h_6(x)$</th>
<th>$h_7(x)$</th>
<th>$h_8(x)$</th>
<th>$h_9(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>sushi</td>
<td>was</td>
<td>great</td>
<td>the</td>
<td>food</td>
<td>awesome</td>
<td>but</td>
<td>service</td>
<td>terrible</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Want to compute the probability of seeing our dataset for every possible setting for $w$. Find $w$ that makes data most likely! (e.g., maximize this likelihood metric)

<table>
<thead>
<tr>
<th>Data Point</th>
<th>$h_1(x)$</th>
<th>$h_2(x)$</th>
<th>$y$</th>
<th>Choose $w$ to maximize</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1, y_1$</td>
<td>2</td>
<td>1</td>
<td>+1</td>
<td>$P(y_1 = +1</td>
</tr>
<tr>
<td>$x_2, y_2$</td>
<td>0</td>
<td>2</td>
<td>−1</td>
<td>$P(y_2 = −1</td>
</tr>
<tr>
<td>$x_3, y_3$</td>
<td>3</td>
<td>3</td>
<td>−1</td>
<td>$P(y_3 = −1</td>
</tr>
<tr>
<td>$x_4, y_4$</td>
<td>4</td>
<td>1</td>
<td>+1</td>
<td>$P(y_4 = +1</td>
</tr>
</tbody>
</table>
Now that we have our new model, we will talk about how to choose $\hat{w}$ to be the “best fit”.

The choice of $w$ affects how likely seeing our dataset is

$$\ell(w) = \prod_{i}^{n} P(y_i | x_i, w)$$

$$P(y_i = +1 | x_i, w) = \frac{1}{1 + e^{-w^T h(x_i)}}$$

$$P(y_i = -1 | x_i, w) = \frac{e^{-w^T h(x_i)}}{1 + e^{-w^T h(x_i)}}$$
Find the $w$ that maximizes the likelihood

$$
\hat{w} = \arg\max_w \ell(w) = \arg\max_w \prod_{i=1}^{n} P(y_i|x_i, w)
$$

Generally, we maximize the log-likelihood which looks like

$$
\hat{w} = \arg\max_w \ell(w) = \arg\max_w \log(\ell(w)) = \arg\max_w \sum_{i=1}^{n} \log(P(y_i|x_i, w))
$$

Also commonly written by separating out positive/negative terms

$$
\hat{w} = \arg\max_w \sum_{i=1: y_i = +1}^{n} \ln\left(\frac{1}{1 + e^{-w^T h(x)}}\right) + \sum_{i=1: y_i = -1}^{n} \ln\left(\frac{e^{-w^T h(x)}}{1 + e^{-w^T h(x)}}\right)
$$
Which setting of $w$ should we use?

\[ \ell(w^{(1)}) = 10^{-5} \]
\[ \ell(w^{(2)}) = 10^{-6} \]
\[ \ell(w^{(3)}) = 10^{-4} \]
Brain Break
Training Data

_pre-processing_

ML model

\( y \)

\( \hat{y} \)

Optimization algorithm

\( \hat{w} \)

Quality metric

\( x \)

\( h(x) \)

\( \hat{w} \)

\( \hat{y} \)
Finding MLE

No closed-form solution, have to use an iterative method.

Since we are maximizing likelihood, we use gradient ascent.

\[
\hat{w} = \arg\max_w \sum_{i=1}^{n} \log(P(y_i|x_i, w))
\]
Gradient ascent is the same as gradient descent, but we go "up the hill".

\[
\begin{align*}
\text{start at some (random) point } w^{(0)} \text{ when } t &= 0 \\
\text{while we haven’t converged} \\
\quad w^{(t+1)} &\leftarrow w^{(t)} + \eta \nabla \log(\ell(w^{(t)})) \\
\quad t &\leftarrow t + 1
\end{align*}
\]

This is just describing going up the hill step by step.

$\eta$ controls how big of steps we take, and picking it is crucial for how well the model you learn does!
Learning Curve

Log likelihood over all data points

# of iterations

step_size = 1.0e-05
Choosing $\eta$

Step-size too small

![Graph showing log likelihood over iterations with two curves, one for step_size=1.0e-05 and the other for step_size=1.0e-06. The blue curve for step_size=1.0e-05 converges faster than the green curve for step_size=1.0e-06.]
Choosing $\eta$

What about a larger step-size?
Choosing $\eta$

What about a larger step-size?

Can cause divergence!

![Graph showing log likelihood over all data points against number of iterations for different step sizes.](image)
Choosing $\eta$

Unfortunately, you have to do a lot of trial and error 😞

Try several values (generally exponentially spaced)

Find one that is too small and one that is too large to narrow search range. Try values in between!

Advanced: Divergence with large step sizes tends to happen at the end, close to the optimal point. You can use a decreasing step size to avoid this

$$\eta_t = \frac{\eta_0}{t}$$
We have introduced yet another hyperparameter that you have to choose, that will affect which predictor is ultimately learned.

If you want to tune both a Ridge penalty and a learning rate (step size for gradient descent), you will need to try all pairs of settings!

For example, suppose you wanted to try using a validation set to select the right settings out of:

- \( \lambda \in [0.01, 0.1, 1, 10, 100] \)
- \( \eta_t \in \left[ 0.001, 0.01, 0.1, 1, \frac{1}{t}, \frac{10}{t} \right] \)

You will need to train 30 different models and evaluate each one!
Brain Break
Overfitting - Classification
More Features

Like with regression, we can learn more complicated models by including more features or by including more complex features.

Instead of just using

\[ h_1(x) = \#\text{awesome} \]
\[ h_2(x) = \#\text{awful} \]

We could use

\[ h_1(x) = \#\text{awesome} \]
\[ h_2(x) = \#\text{awful} \]
\[ h_3(x) = \#\text{awesome}^2 \]
\[ h_4(x) = \#\text{awful}^2 \]
\[ \ldots \]
Decision Boundary

\[ w^T h(x) = 0.23 + 1.12x[1] - 1.07x[2] \]

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
<th>Coefficient learned</th>
</tr>
</thead>
<tbody>
<tr>
<td>( h_0(x) )</td>
<td>1</td>
<td>0.23</td>
</tr>
<tr>
<td>( h_1(x) )</td>
<td>( x[1] )</td>
<td>1.12</td>
</tr>
<tr>
<td>( h_2(x) )</td>
<td>( x[2] )</td>
<td>-1.07</td>
</tr>
</tbody>
</table>
Decision Boundary

\[ w^T h(x) = 1.68 + 1.39x[1] - 0.59x[2] - 0.17x[1]^2 - 0.96x[2]^2 \]

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
<th>Coefficient learned</th>
</tr>
</thead>
<tbody>
<tr>
<td>( h_0(x) )</td>
<td>1</td>
<td>1.68</td>
</tr>
<tr>
<td>( h_1(x) )</td>
<td>( x[1] )</td>
<td>1.39</td>
</tr>
<tr>
<td>( h_2(x) )</td>
<td>( x[2] )</td>
<td>-0.59</td>
</tr>
<tr>
<td>( h_3(x) )</td>
<td>( (x[1])^2 )</td>
<td>-0.17</td>
</tr>
<tr>
<td>( h_4(x) )</td>
<td>( (x[2])^2 )</td>
<td>-0.96</td>
</tr>
</tbody>
</table>
Decision Boundary

\[ w^T h(x) = \ldots \]

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
<th>Coefficient learned</th>
</tr>
</thead>
<tbody>
<tr>
<td>( h_0(x) )</td>
<td>1</td>
<td>21.6</td>
</tr>
<tr>
<td>( h_1(x) )</td>
<td>( x[1] )</td>
<td>5.3</td>
</tr>
<tr>
<td>( h_2(x) )</td>
<td>( x[2] )</td>
<td>-42.7</td>
</tr>
<tr>
<td>( h_3(x) )</td>
<td>( (x[1])^2 )</td>
<td>-15.9</td>
</tr>
<tr>
<td>( h_4(x) )</td>
<td>( (x[2])^2 )</td>
<td>-48.6</td>
</tr>
<tr>
<td>( h_5(x) )</td>
<td>( (x[1])^3 )</td>
<td>-11.0</td>
</tr>
<tr>
<td>( h_6(x) )</td>
<td>( (x[2])^3 )</td>
<td>67.0</td>
</tr>
<tr>
<td>( h_7(x) )</td>
<td>( (x[1])^4 )</td>
<td>1.5</td>
</tr>
<tr>
<td>( h_8(x) )</td>
<td>( (x[2])^4 )</td>
<td>48.0</td>
</tr>
<tr>
<td>( h_9(x) )</td>
<td>( (x[1])^5 )</td>
<td>4.4</td>
</tr>
<tr>
<td>( h_{10}(x) )</td>
<td>( (x[2])^5 )</td>
<td>-14.2</td>
</tr>
<tr>
<td>( h_{11}(x) )</td>
<td>( (x[1])^6 )</td>
<td>0.8</td>
</tr>
<tr>
<td>( h_{12}(x) )</td>
<td>( (x[2])^6 )</td>
<td>-8.6</td>
</tr>
</tbody>
</table>
Decision Boundary

\[ w^T h(x) = \ldots \]

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
<th>Coefficient learned</th>
</tr>
</thead>
<tbody>
<tr>
<td>( h_0(x) )</td>
<td>1</td>
<td>8.7</td>
</tr>
<tr>
<td>( h_1(x) )</td>
<td>( x[1] )</td>
<td>5.1</td>
</tr>
<tr>
<td>( h_2(x) )</td>
<td>( x[2] )</td>
<td>78.7</td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( h_{11}(x) )</td>
<td>( (x[1])^6 )</td>
<td>-7.5</td>
</tr>
<tr>
<td>( h_{12}(x) )</td>
<td>( (x[2])^6 )</td>
<td>3803</td>
</tr>
<tr>
<td>( h_{13}(x) )</td>
<td>( (x[1])^7 )</td>
<td>21.1</td>
</tr>
<tr>
<td>( h_{14}(x) )</td>
<td>( (x[2])^7 )</td>
<td>-2406</td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( h_{37}(x) )</td>
<td>( (x[1])^{19} )</td>
<td>-2*10^{-6}</td>
</tr>
<tr>
<td>( h_{38}(x) )</td>
<td>( (x[2])^{19} )</td>
<td>-0.15</td>
</tr>
<tr>
<td>( h_{39}(x) )</td>
<td>( (x[1])^{20} )</td>
<td>-2*10^{-6}</td>
</tr>
<tr>
<td>( h_{40}(x) )</td>
<td>( (x[2])^{20} )</td>
<td>0.03</td>
</tr>
</tbody>
</table>
Just like with regression, we see a similar pattern with complexity.

Classical Error

Train Error

Low Complexity

High Complexity

True Error
Remember, we say the logistic function become “sharper” with larger coefficients.

<table>
<thead>
<tr>
<th>w_0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>w_{#awesome}</td>
<td>+1</td>
</tr>
<tr>
<td>w_{#awful}</td>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>w_0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>w_{#awesome}</td>
<td>+2</td>
</tr>
<tr>
<td>w_{#awful}</td>
<td>-2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>w_0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>w_{#awesome}</td>
<td>+6</td>
</tr>
<tr>
<td>w_{#awful}</td>
<td>-6</td>
</tr>
</tbody>
</table>

What does this mean for our predictions?

Because the \( \text{Score}(x) \) is getting larger in magnitude, the probabilities are closer to 0 or 1!
Plotting Probabilities

\[ P(y = +1|x) = \frac{1}{1 + e^{-\omega^T h(x)}} \]
Regularization
L2 Regularized Logistic Regression

Just like in regression, can change our quality metric to avoid overfitting when training a model

$$\hat{w} = \arg \max_w \log(\ell(w)) - \lambda \|w\|^2_2$$

<table>
<thead>
<tr>
<th>Regularization</th>
<th>$\lambda = 0$</th>
<th>$\lambda = 0.00001$</th>
<th>$\lambda = 0.001$</th>
<th>$\lambda = 1$</th>
<th>$\lambda = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range of coefficients</td>
<td>-3170 to 3803</td>
<td>-8.04 to 12.14</td>
<td>-0.70 to 1.25</td>
<td>-0.13 to 0.57</td>
<td>-0.05 to 0.22</td>
</tr>
<tr>
<td>Decision boundary</td>
<td><img src="image1.png" alt="Decision boundary" /></td>
<td><img src="image2.png" alt="Decision boundary" /></td>
<td><img src="image3.png" alt="Decision boundary" /></td>
<td><img src="image4.png" alt="Decision boundary" /></td>
<td><img src="image5.png" alt="Decision boundary" /></td>
</tr>
<tr>
<td>Learned probabilities</td>
<td><img src="image1.png" alt="Learned probabilities" /></td>
<td><img src="image2.png" alt="Learned probabilities" /></td>
<td><img src="image3.png" alt="Learned probabilities" /></td>
<td><img src="image4.png" alt="Learned probabilities" /></td>
<td><img src="image5.png" alt="Learned probabilities" /></td>
</tr>
</tbody>
</table>
Why do we subtract the L2 Norm?

\[ \hat{w} = \arg\max_w \log(\ell(w)) - \lambda \|w\|_2^2 \]

How does \( \lambda \) impact the complexity of the model?

How do we pick \( \lambda \)?
Coefficient Path: L2 Penalty

Coefficient $\hat{w}_j$
Jake wants to find the best Logistic Regression model for a sentiment analysis dataset by tuning the regularization parameter $\lambda \in [0, 10^{-2}, 10^{-1}, 1, 10]$ and the learning rate $\eta \in [10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}]$. He does the following:

- Runs cross-validation on $\lambda$ to get the best value for the regularization parameter.
- For that value of $\lambda$, run cross-validation on $\eta$ to get the best value for the learning rate.

After running this procedure, he is convinced he has the best Logistic Regression model for his dataset, given the hyper-parameter values he wanted to test.

**What did Jake do wrong?**
Recap

**Theme**: Details of logistic classification and how to train it

**Ideas:**
- Predict with probabilities
- Using the logistic function to turn Score to probability
- Logistic Regression
- Minimizing error vs maximizing likelihood
- Gradient Ascent
- Effects of learning rate
- Overfitting with logistic regression
  - Over-confident (probabilities close to 0 or 1)
  - Regularization