Exceptions to the Rule

A gentle introduction to ML
Recap: Simulation v.s. Model Fitting

\[ y_i = f_\theta(x_i, \epsilon_i) \]

**Simulation**: Observe \( x_i, e_i, \) and \( \theta \), generate \( y_i \)

**Fitting**: Observe \( x_i, y_i \), infer \( \theta \)

Why not observe \( \epsilon_i \)?

“Generic” term that captures process/observation/modeling error
Recap: Bias-Variance Tradeoff

**Bias:** How closely does your selected model align with the data.

**Variance:** How different predictions are (are nearby predictions similar to each other).
Recap: Bias-Variance Tradeoff
Recap: Bias-Variance Tradeoff

Large classes (more parameters) have lower bias but higher variance.
Recap: Bias-Variance Tradeoff

Degree of polynomial of the fit
Recap: Training and Testing

- Use this data for fitting
- Use this data for evaluation
Recap: Training and Testing

**Training error**: average loss over the training set

**Testing error**: average loss over the testing set
Recap: Training and Testing

![Graph showing the relationship between model complexity and error in training and testing. The graph illustrates the trade-off between underfitting and overfitting. As the model complexity increases, the training error decreases, but the test error increases. There is a point of best fit where the model complexity is balanced, minimizing both training and test errors.](image-url)
Another Perspective
Another Perspective

What patterns do we see in this data?
Another Perspective

<table>
<thead>
<tr>
<th>Got Sick?</th>
<th>Will I Get Sick?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

Both “sick” fruits are round
Another Perspective

Both “not sick” fruits are yellow
Another Perspective

What about \((\text{is\_oval} \ \text{AND} \ \text{not\_yellow})\)?
Overfitting

Rules are too precise

Got Sick?
Y
N
Y
N

(Purple And Spots) OR (Orange With Leaves)

Unlikely to be relevant on new fruits
Underfitting

Rules are not expressive enough

Unlikely to be relevant on new fruits
Overfitting/Underfitting

Model Complexity

How many ways can the rules partition the data

Not expressive enough

Does not apply to future data
Same Tradeoff!
Training and Testing

Original Data → Randomly Selected Rows → Training Data

Use this data for rule selection

Training Data → Testing Data

Use this data for rule evaluation
More Formally

<table>
<thead>
<tr>
<th>Features</th>
<th>X</th>
<th>Y (got sick?)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[redness, how oval]</td>
<td>[0.95, 0.25]</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>[0.25, 0.85]</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>[0.45, 0.05]</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>[0.05, 0.75]</td>
<td>0</td>
</tr>
</tbody>
</table>
Another Perspective

<table>
<thead>
<tr>
<th>X</th>
<th>Y (got sick?)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[redness, how oval]</td>
<td></td>
</tr>
<tr>
<td>[0.95, 0.25]</td>
<td>1</td>
</tr>
<tr>
<td>[0.25, 0.85]</td>
<td>0</td>
</tr>
<tr>
<td>[0.45, 0.05]</td>
<td>1</td>
</tr>
<tr>
<td>[0.05, 0.75]</td>
<td>0</td>
</tr>
</tbody>
</table>

Find a function $f(x) \rightarrow y$
Exactly the same

\[ y_i = f_\theta(x_i, \epsilon_i) \]

**Fitting**: Observe xi, yi, infer theta
Another Perspective

redness

[0.45, 0.05]

[0.05, 0.75]

[0.95, 0.25]

ovalness

[0.25, 0.85]
Another Perspective

- Redness: [0.95, 0.25]
- Ovalness: [0.45, 0.05], [0.05, 0.75], [0.25, 0.85]
Another Perspective

- redness
- ovalness
High Bias

Redness

Ovalness

Sick

Not Sick
Low Bias

![Graph with redness and ovalness axes showing 'Sick' and 'Not Sick' categories with coordinates (0.95, 0.25) and (0.25, 0.85)]
High Variance

- **Redness**: [0.45, 0.05] and [0.05, 0.75]
- **Ovalness**: [0.25, 0.85]
Low Variance

- redness: [0.45, 0.05], [0.05, 0.75]
- ovalness: [0.25, 0.85]
What Does the Model Look Like?

\[ y_i = f_{\theta}(x_i, \epsilon_i) \]

**Fitting:** Observe \( x_i, y_i \), infer \( \theta \)
“Logistic” Regression

Function with values between 0 and 1

\[ S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}. \]
“Logistic Regression”

\[ f(x) = \theta^T x \]

- Linear function of the features.

\[ g(x) = \frac{1}{1 + e^{-f(x)}} \]

- When \( f(x) \) is high (greater than 0), \( g(x) \) tends towards 1
- When \( f(x) \) is low (less than 0), \( g(x) \) tends towards 0
A Linear Partitioning

\[ \theta x > 0.5 \]

\[ \theta x < 0.5 \]

Closer to 1

Closer to 0
# Objective

<table>
<thead>
<tr>
<th>X</th>
<th>Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>[redness, how oval]</td>
<td></td>
</tr>
<tr>
<td>[0.95, 0.25]</td>
<td>0.96 ~~ 1</td>
</tr>
<tr>
<td>[0.25, 0.85]</td>
<td>0.1 ~~ 0</td>
</tr>
<tr>
<td>[0.45, 0.05]</td>
<td>0.87 ~~ 1</td>
</tr>
<tr>
<td>[0.05, 0.75]</td>
<td>0.14 ~~ 1</td>
</tr>
</tbody>
</table>

Get the predictions as close to true 0 or 1
“Logistic Regression”

Loss function

\[ L(y_i, g_\theta(x_i)) = \begin{cases} 
-\log(g_\theta(x_i)) & \text{if } y_i = 1 \\
-\log(1 - g_\theta(x_i)) & \text{if } y_i = 0 
\end{cases} \]
"Logistic Regression"

```python
>>> from sklearn.datasets import load_iris
>>> from sklearn.linear_model import LogisticRegression
>>> X, y = load_iris(return_X_y=True)
>>> clf = LogisticRegression(random_state=0).fit(X, y)
>>> clf.predict(X[2, :])
array([0, 0])
>>> clf.predict_proba(X[2, :])
array([[9.8...e-01, 1.8...e-02, 1.4...e-08],
       [9.7...e-01, 2.8...e-02, ...e-08]])
>>> clf.score(X, y)
0.97...
```
A Linear Partitioning

redness

ovalness

[0.45, 0.05] Closer to 0

[0.95, 0.25] Closer to 1

0.5

[0.05, 0.75]

[0.25, 0.85]
“Linear” Regression

Least squares principle

\[ \min_{a,c} \sum_{i=1}^{N} (y_i - (ax_i + c))^2 \]

Find parameters that minimize the squared error between prediction and actual.
"Linear” Regression

Least squares principle

\[
\min_{\theta} \sum_{i=1}^{N} (y_i - \theta^T x)^2
\]

Find parameters that minimize the squared error between prediction and actual.
“Logistic” Regression

Least squares principle (kind of...)

$$\min_{\theta} \sum_{i=1}^{N} -y_i \log(g_{\theta}(x_i)) + (1 - y_i) \log(1 - g_{\theta}(x_i))$$

Find parameters that linearly partition the feature space to best match the labels.
Recap: Simulation v.s. Model Fitting

\[ y_i = f_\theta(x_i, \epsilon_i) \]

**Simulation**: Observe \( x_i, e_i, \) and \( \theta \), **generate** \( y_i \)

**Fitting**: Observe \( x_i, y_i \), **infer** \( \theta \)

Regression Problems -> Continuous Y

Classification Problems -> Discrete Y
Machine Learning

Given features (X) and labels (Y)
Find a function to predict Y from X
Features Are Up-to You

Features

- redness, how oval

<table>
<thead>
<tr>
<th>Features</th>
<th>Y (got sick?)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0.95, 0.25]</td>
<td>1</td>
</tr>
<tr>
<td>[0.25, 0.85]</td>
<td>0</td>
</tr>
<tr>
<td>[0.45, 0.05]</td>
<td>1</td>
</tr>
<tr>
<td>[0.05, 0.75]</td>
<td>0</td>
</tr>
</tbody>
</table>