GEOMETRIC VIEW OF DATA

Proper Experimentation

REAL WORLD USE OF ML ALGORITHMS

past
Training Data (data with labels)

future
Testing Data (data without labels)

How do we tell how well we’re doing?
### Real-world classification

Google has labeled training data, for example from people clicking the "spam" button, but when new messages come in, they’re not labeled.

### Classification evaluation

<table>
<thead>
<tr>
<th>Data</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Use the labeled data we have already to create a test set with known labels!

**Why can we do this?**

We assume there’s an underlying distribution that generates both the training and test examples.

### Classification evaluation

<table>
<thead>
<tr>
<th>Data</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Train a classifier

### Classification evaluation

<table>
<thead>
<tr>
<th>Data</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Train a classifier
Classification evaluation

Data | Label
--- | ---
1 | 0

Pretend like we don’t know the labels

Classify

Compare predicted labels to actual labels

How could we score these for classification?

To evaluate the model, compare the predicted labels to the actual labels

Accuracy: the proportion of examples where we correctly predicted the label
One way to do algorithm development:
- try out an algorithm
- evaluate on test data
- repeat until happy with results

Is this ok?

No. Although we’re not explicitly looking at the examples, we’re still “cheating” by biasing our algorithm to the test data.

Once you look at/use test data it is no longer test data!

So, how can we evaluate our algorithm during development?

Using the development data:
- try out an algorithm
- evaluate on development data
- repeat until happy with results

When satisfied, evaluate on test data.
Proper testing

- Training Data
- Development Data

- Learn
- Evaluate model
- Any problems with this?

Overfitting to development data

- Be careful not to overfit the development data!

- All Training Data
- Training Data
- Development Data

Often we'll split off development data multiple times (in fact, on the fly)... you can still overfit, but this helps avoid it

Pruning revisited

- Unicycle
- Mountain
- Normal
- Terrain
- YES
- Trail
- NO
- Rainy
- Snowy
- Sunny
- NO

Which should we pick?

Pruning revisited

- Unicycle
- Mountain
- Normal
- Terrain
- YES
- Trail
- NO
- Rainy
- Snowy
- Sunny
- NO

Use development data to decide!
Can we visualize this data?

We can view examples as points in an $n$-dimensional space where $n$ is the number of features.
Another classification algorithm?

To classify an example \(d\):
- Label \(d\) with the label of the closest example to \(d\) in the training set.
What about this example?

closest to red, but...

Most of the next closest are blue

$k$-Nearest Neighbor (k-NN)

To classify an example $d$:
- Find $k$ nearest neighbors of $d$
- Choose as the label the majority label within the $k$ nearest neighbors

How do we measure "nearest"?
Euclidean distance

In two dimensions, how do we compute the distance?

\[ D(a, b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2} \]

Euclidean distance

In n-dimensions, how do we compute the distance?

\[ D(a, b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2} \]

Euclidean distance

In n-dimensions, how do we compute the distance?

Measuring distance/similarity is a domain-specific problem and there are many, many different variations!

Decision boundaries

The decision boundaries are places in the feature space where the classification of a point/example changes.

Where are the decision boundaries for k-NN?
**k-NN decision boundaries**

- k-NN gives locally defined decision boundaries between classes.

---

**Choosing k**

- What is the label with $k = 1$?

---

**Choosing k**

- We'd choose red. Do you agree?

---

**Choosing k**

- What is the label with $k = 3$?
Choosing $k$

We'd choose blue. Do you agree?

What is the label with $k = 100$?

Choosing $k$

We'd choose red. Do you agree?

The impact of $k$

What is the role of $k$?
How does it relate to overfitting and underfitting?
How did we control this for decision trees?
To classify an example $d$:
- Find $k$ nearest neighbors of $d$
- Choose as the class the majority class within the $k$ nearest neighbors

How do we choose $k$?

How to pick $k$

Common heuristics:
- often 3, 5, 7
- choose an odd number to avoid ties
Use development data

k-NN variants

To classify an example $d$:
- Find $k$ nearest neighbors of $d$
- Choose as the class the majority class within the $k$ nearest neighbors

Any variation ideas?

k-NN variations

Instead of $k$ nearest neighbors, count majority from all examples within a fixed distance

Weighted k-NN:
- Right now, all examples are treated equally
- weight the “vote” of the examples, so that closer examples have more vote/weight
- often use some sort of exponential decay
Decision boundaries for decision trees

What do the decision boundaries for decision trees like?

Axis-aligned splits/cuts of the data

What types of data sets will DT work poorly on?

Problems for DT
Decision trees vs. k-NN

- Which is faster to train?
  - k-NN doesn’t require any training!

- Which is faster to classify?
  - For most data sets, decision trees

- Do they use the features in the same way to label the examples?
  - k-NN treats all features equally! Decision trees “select” important features

Machine learning models

- Some machine learning approaches make strong assumptions about the data
  - If the assumptions are true, it can often lead to better performance
  - If the assumptions aren’t true, the approach can fail miserably

- Other approaches don’t make many assumptions about the data
  - This can allow us to learn from more varied data
  - But, they are more prone to overfitting
  - and generally require more training data

Data generating distribution

- We are going to use the probabilistic model of learning

- There is some probability distribution over example/label pairs called the data generating distribution

  - Both the training data and the test set are generated based on this distribution

  - What is a probability distribution?
Probability distribution

- Describes how likely (i.e. probable) certain events are
- Describes probabilities for all possible events
- Probabilities are between 0 and 1 (inclusive)
- Sum of probabilities over all events is 1

Data generating distribution

- Training data
- Test set
- Data generating distribution

What is the data generating distribution?
What is the data generating distribution?
If you don’t have strong assumptions about the model, it can take you a longer to learn. Assume now that our model of the blue class is two circles.

What is the data generating distribution?
What is the data generating distribution?

Actual model
What is the data generating distribution?

Knowing the model beforehand can drastically improve the learning and the number of examples required.

Make sure your assumption is correct, though!

Machine learning models

- What are the model assumptions (if any) that k-NN and decision trees make about the data?
- Are there data sets that could never be learned correctly by either?
k-NN model

No model assumptions. Assumes that proximity relates to class.

Decision tree model

Axis-aligned splits/cuts of the data.

Bias

The "bias" of a model is how strong the model assumptions are.

Low-bias classifiers make minimal assumptions about the data (k-NN and DT are generally considered low bias).

High-bias classifiers make strong assumptions about the data.

Linear models

A strong high-bias assumption is linear separability:

- In 2 dimensions, can separate classes by a line
- In higher dimensions, need hyperplanes

A linear model is a model that assumes the data is linearly separable.