Proper Experimentation

How do we tell how well we’re doing?

REAL WORLD USE OF ML ALGORITHMS

Training Data (data with labels) → Learn → Testing Data (data without labels) → Predict
**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**

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

Why can we do this?

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

---

**Classification evaluation**

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

**Classification evaluation**

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

---

**Classification evaluation**

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

---

**Classification evaluation**

Data | Label
--- | ---
0 | 0
0 | 1
1 | 0
1 | 1
0 | 0
Classification evaluation

Pretend like we don’t know the labels

Classify

Compare predicted labels to actual labels

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

Accuracy: the proportion of examples where we correctly predicted the label
Proper testing

One way to do algorithm development:
- try out an algorithm
- evaluated 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.

Development set

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

When satisfied, evaluate on test data

Evaluate model
Proper testing

Training Data → learn → Development Data → Evaluate model

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

Any problems with this?

Overfitting to development data

Be careful not to overfit to the development data!

All 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

Which should we pick?

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

Turn features into numerical values (read the book for a more detailed discussion of this).

We can view examples as points in an $n$-dimensional space where $n$ is the number of features.
Test example: *what class?*

- Feature 1
- Feature 2
- Closest to red

Test example: *what class?*

- Feature 1
- Feature 2
- Closest to red

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 his example?

- Feature 1
- Feature 2
What about his example?

What about his example?

What about his example?

What about his example?

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

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?

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

Decision boundaries

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

Where are the decision boundaries for k-NN?

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

What is the label with $k=1$?

We'd choose red. Do you agree?

What is the label with $k=3$?

We'd choose blue. Do you agree?
Choosing $k$

What is the label with $k = 100$?

feature$_1$

feature$_2$

label 1
label 2
label 3

We’d choose blue. Do you agree?

feature$_1$

feature$_2$

label 1
label 2
label 3

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?

$k$-Nearest Neighbor ($k$-NN)

To classify an example $d$:
1. Find $k$ nearest neighbors of $d$
2. 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 are the decision boundaries for decision trees like?
Decision boundaries for decision trees

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?

Which is faster to classify?

Do they use the features in the same way to label the examples?
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 this can often lead to better performance
  - If the assumptions aren't true, they 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
What is the data generating distribution?
If you don’t have strong assumptions about the model, it can take you 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?

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

What is the data generating distribution?

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?
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.

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.
Hyperplanes

A hyperplane is line/plane in a high dimensional space

What defines a line?
What defines a hyperplane?

Defining a line

Any pair of values \((w_1, w_2)\) defines a line through the origin:

\[ 0 = w_1 f_1 + w_2 f_2 \]
Defining a line

Any pair of values \((w_1, w_2)\) defines a line through the origin:

\[ 0 = w_1 f_1 + w_2 f_2 \]

\[ 0 = 1f_1 + 2f_2 \]

\[ w = (1, 2) \]

We can also view it as the line perpendicular to the weight vector

Classifying with a line

Mathematically, how can we classify points based on a line?

\[ 0 = 1f_1 + 2f_2 \]

The sign indicates which side of the line

Classifying with a line

Mathematically, how can we classify points based on a line?

\[ 0 = 1f_1 + 2f_2 \]

\[ (1,1): 1 * 1 + 2 * 1 = 3 \]

\[ (1,-1): 1 * 1 + 2 * -1 = -1 \]

\[ w = (1, 2) \]

The sign indicates which side of the line

Defining a line

Any pair of values \((w_1, w_2)\) defines a line through the origin:

\[ 0 = w_1 f_1 + w_2 f_2 \]

\[ 0 = 1f_1 + 2f_2 \]

\[ w = (1, 2) \]

How do we move the line off of the origin?
Defining a line

Any pair of values \((w_1, w_2)\) defines a line through the origin:

\[ a = w_1 f_1 + w_2 f_2 \]

\[-1 = 1 f_1 + 2 f_2\]

\[ \begin{array}{c}
-2 \\
-1 \\
0 \\
1 \\
2 \\
\end{array} \]

Now intersects at -1

Linear models

A linear model in \(n\)-dimensional space (i.e. \(n\) features) is defined by \(n+1\) weights:

In two dimensions, a line:

\[ 0 = w_1 f_1 + w_2 f_2 + b \quad \text{(where } b \leq 0) \]

In three dimensions, a plane:

\[ 0 = w_1 f_1 + w_2 f_2 + w_3 f_3 + b \]

In \(n\)-dimensions, a hyperplane

\[ 0 = b + \sum_{i=1}^{n} w_i f_i \]

Classification with a linear model

We can classify with a linear model by checking the sign:

\[ f_n, f_0, \ldots, f_0 \]

\[ b + \sum_{i=1}^{n} w_i f_i > 0 \quad \text{Positive example} \]

\[ b + \sum_{i=1}^{n} w_i f_i < 0 \quad \text{Negative example} \]
An aside: a thought experiment

What is a 100,000-dimensional space like?
You’re a 1-D creature, and you decide to buy a 2-unit apartment

2 rooms (very, skinny rooms)

Another thought experiment

What is a 100,000-dimensional space like?
Your job’s going well and you’re making good money. You upgrade to a 2-D apartment with 2-units per dimension

4 rooms (very, flat rooms)

Another thought experiment

What is a 100,000-dimensional space like?
You get promoted again and start having kids and decide to upgrade to another dimension.

8 rooms (very, normal rooms)

Each time you add a dimension, the amount of space you have to work with goes up exponentially.

Another thought experiment

What is a 100,000-dimensional space like?
Larry Page steps down as CEO of google and they ask you if you’d like the job. You decide to upgrade to a 100,000 dimensional apartment.

How much room do you have? Can you have a big party?

$2^{100,000}$ rooms (it’s very quiet and lonely…) = $10^{30}$ rooms per person if you invited everyone on the planet
The challenge

Our intuitions about space/distance don't scale with dimensions!