Explanatory Notes for 6.390

Shaunticlair Ruiz (Current TA)

Fall 2022

Evaluating our Learning Algorithm

So, while we can evaluate each **hypothesis**, it's also important to measure how our **learning algorithm** is performing.

How do we measure it? Well, the job of our **learning algorithm** is to **pick good hypotheses**.

Concept 1

We can **evaluate** the performance of a **learning algorithm** using **testing loss**: a good learning algorithm will create **hypotheses** with **low testing loss**.

You could think of this as measuring the **skill** of a **teacher** (the learning algorithm) by the **success** of their **student** (the hypothesis) on a **test** (testing loss).

Validation: Evaluating with lots of data

When we were creating hypotheses, **randomness** caused some problems: you might not get **training data** that matched the **testing data** very well.

The **same** can happen here, when **evaluating** your **algorithm**: maybe your model happened to create a bad (or unusually good!) hypothesis because of **luck**.

The easy solution to **randomness** is to add **more data**: we get more **consistency** that way.

So, we **repeatedly** get new training data and test data. For each, we **train** a different hypothesis. We can **average** their performance out, and use that to **estimate** the quality of our algorithm.

Definition 2

Validation is a way to **evaluate a learning algorithm** using **large amounts of data**.

We do this by **running** our algorithm **many times** with new data, and **averaging** the testing error of all the hypotheses.

This process is often requires having **lots of data** to train with, but is a **provably** good approach.

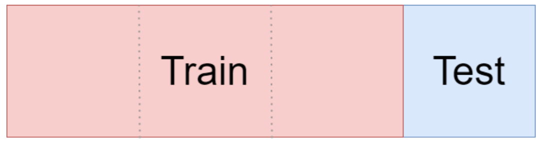
Our Problem: When data is less available

As mentioned, this takes up **lots of data**. What if we can't get as much: it's **expensive**, or not even possible? In this case, we have some **finite** data, $\mathcal{D}_n$. We **can't get more**.

We solved the **randomness** problem by using **more** training and testing **data**. So, we need some way to **get** more **distinct** hypotheses.

One set of data gives us one **hypothesis**. But, what if, rather than using **completely** new data, we used **slightly different** data each time?

First, need to break $\mathcal{D}_n$ into a chunk for training, and a chunk for testing.



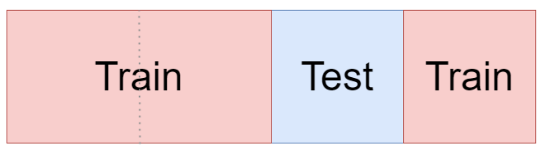
How do we get more hypotheses from this dataset?

Cross-Validation

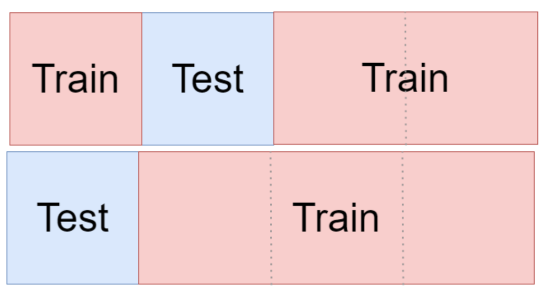
We mentioned that we just want **different** hypotheses. Our hypotheses depend on our **training data**. So we want to **change** our training data.

We can't **add** data to it, because then we **lose** testing data. We shouldn't **remove** data, because then we're just making a hypothesis that's **less well-informed**.

Instead, we'll **swap** some of the training data for testing data.



This will create a new hypothesis, and the data is partially different! In fact, we can do this for each of our chunks:



We now have **four different hypotheses** for the price of one!

Definition 3

Cross-validation is a way to **evaluate a learning algorithm** using **limited data**.

We do this by **breaking** our data it into **chunks** to create **multiple hypotheses** from one dataset.

For each **chunk**, we train one dataset on all the data **not in that chunk**. We get our **test error** using the chunk **we left out**.

For k chunks, we end up with k hypotheses. By **averaging** out their performance, we can **approximate** the quality of our algorithm.

This approach is much **less expensive**, and very common in machine learning! But, some of the theoretical **benefits** of validation are not **proven** to be true for cross-validation.

Clarification 4

Note that the goal of validation and cross-validation is **not** to evaluate **one hypothesis**.

Instead, it is instead meant to evaluate a **learning algorithm**. This is why we have to create **many** hypotheses: we want to see that our algorithm is **generally** good!