Table of Contents:

## Problem

Problem: To find the underlying function \(f:X \mapsto Y\) that maps from input \(X\) to output \(Y\) such that for any \(x \in X\) the prediction error is low. Example,

## Basic Supervised ML Problems

- Classification
  - Predict which class an example belongs to
  - E.g. spam filtering example

$$f(x|w,b) = sign(w^T - b)$$

- Regression
  - Predict a real value or a probability
  - E.g., probability of being spam

$$f(x|w,b) = w^T - b$$

- Highly inter-related
  - Train on Regression => Use for classification

## [Example] Spam Filtering

Why is Spam Filtering Hard ?

1. Easy for humans to recognise
2. Hard for humans to write the algorithm
3. Lots of *if.. else..* statements

One way to solve this problem is by *Bags of Words (aka Feature Vector)* where the model prodicts whether the mail is spam based on the keywords used in the mail.

Let \(x\) denote the bag-of-words for the email, e.g. \(x = (1,1,0,0,1,1)\) & output \( y \in [-1, 1] \) denote the mail being spam & not-spam. Then, the learning problem can be formulated as a *classification problem* that can be solved using *linear classifier models*.

## [Model] Linear Classifier model

Linear Classifier model can be represented as:

$$f(x|w,b) = sign(w^T x - b) = sign(w_1 * x_1 + ...w_6 * x_6 - b)$$

Finding the optimal parameters \( (w, b) \) will give us the prediction model for spam filtering problem.

But how to quantify the prediction accuracy ? *Loss Functions* !!

A loss function can be as simple as **0/1 Loss** or **Squared Loss**.

## [Problem Formulation] Basic Supervised Learning Recipe

## [Approach] Gradient Descent

How to find the model paramters ? We will visit this section later.

## [Analysis] Overfitting Vs Underfitting

(A model analysis technic)

So, the steps we have so far are:

1. collect training data
2. train a model on training data
3. test the model

But on testing, the car crashed !! Why ? Turns out, all the training data was collected on the roads with grass on both sides. Now, on the roads, the model didn't know what to do, as it was not trained for such scenerios. This happened because, we *overfitted* the model on the training set that was not actually representative of the true testings. So, minimizing loss on the training set does not garuantee that model will perform well on testing set as well.

### Test Error

We have collected a finite set of samples from the training dataset \( S \) independent of the true distribution \( P \) (which is same as test set but unknown to us). Collecting training data (collect, filtering, labelling, ...) is expensive for both time & financial reasons. We try our best to collect the data on true distribution but we never know the distribution.

**Test Error**: The error that the model makes on the true distribution.

> Overfitting occurs when error on testing set is much higher than on the training set.
> Underfitting occurs when training and test error are both similar and high.

What we can do is, compute the *Expected Test Error*.

### Expected Test Error

**Expected Test Error:** average error when sampled on the distribution $$P$$ rather than the error on single sample \( L(y, f(x))\).

We take out small random set (aka *validation set*) out of the training data set and compute the test error.

> Ideally, for infinite sized dataset this validation set will be the true representation of the test data set.

So, how do we quantitatively find overfitting ?

## Bias-Variance Decomposition

For the Squared loss function, the Expected test error is sum of

1. Variance Term: average squared distance between the prediction of learnt model $$ f_s(x) $$ verses the average prediction $$ F(x) $$. How do we compute $$ F(x) $$ ??
2. Bias Term : The difference between the average prediction of all the model on the validation set $$ F(x) $$ relative to the ground truth $$ y $$.

So, we analyze which of the two terms dominate the test error.

Lets analyze Linear & Quadratic models on simple 2D examples:

The [Train Samples] are randomly choosen from the [Test Samples] . Note that, we use only 7/100 samples for training to make the overfitting problem obvious.

Observations:

1. Model varies significantly based on the training dataset (Variance)
2. Model doesn't fit perfectly to all samples (Bias)

**Overfit vs Underfit**

If we repeat this experiment 1000 times & find the extreme values (min & max) at each input x, we get the variance as shown below for linear, quadratic & cubic models:

Variance is sum of squared distance between the model prediction $$f(x_i)$$ and average prediction of samples from $$P$$ at $$x_i$$. Say among 1000 samples, we have 5 non-zero samples for $$x_i$$ where $$f(x_i) = [0.1, 0.3, -0.2, 0.1, -0.1]$$, then $$F(x_i) = (0.1+0.3-0.2+0.1-0.1)/5 = 0.04$$.

Bias is the vertical distance between the model prediction $$f(x_i)$$ & labels $$ y $$.

(input $$x$$ on the X-axis of both subplots)

Observations:

1. Linear model doesn't fit very well & has high bias => Underfit !
2. Cubic model has great fit but high variance => Overfit !
3. Quadratic model has both low variance and bias => good model !!

> Variance is a measure of the stability of the model and Bias is measure of fit of the model.

Conclusion,

> **High Variance implies overfitting**

- increases with model complexity => implies instability => Overfitting
- decreases with more training data

> **High Bias implies underfitting**

- High error value, even with no variance => model doesn't fit the data well => underfitting
- decreases with model complexity

## Model Selection

- We only have a finite training dataset
- We cannot measure the true test error!
- Bias-variance Trade off
  - Simple model classes underfit
  - Complex model classes overfit

**Goal**: Select the model class with the lowest test error.

## How to estimate test error better way ?

**k-fold Cross Validation**

## Supervised Learning Pipeline

## References

1. Lecture Video 1, Caltech CS155, Winter 2017
2. Caltech CS155, 2022 (new slides)