CS229 and CS3244 Machine Learning AY24/25, ST Notes

Sim Ray En Ryan

December 3, 2025

Contents

1 Introduction

Machine learning is the main driver of recent successes in AI, and moves the complexity of computations from the code to the data. It has an input (a collection), and encoding of that input, a model (predictor) that describes the relationships within and among the examples, and an estimation that will learn (estimate) parameters about the input, and predict outputs, given new outputs.

1.1 Types of Learning

- Supervised: Provided training data with labels
- Unsupervised: Provided training data without labels, find patterns by itself
- Self-Supervised: Use part of the input as supervised training
- Reinforcement Learning: Learning from interactions with the environment

1.2 Supervised Learning

A model is a function from X to Y. For example, $X = \{0, ..255\}^{10000}$ may represent 10000 ASCII characters ina text file, and $Y = \{0, 1\}$ is a boolean output, such as if it is spam or not spam. If the output Y is continuous, it is a regression problem. If it is discrete, it is a classification (or binary, or multiclass) problem.

In machine learning, the training set is a set of pairs of the input and the output, with a size of n. The goal of ML is to use this training set to find a good model for f, and to use that on new data.

2 Linear Regression and Gradient Descent

2.1 Regression

Same as "best fit". Note that Linear Regression is linear in the parameters.

2.1.1 Model

Our assumption about how the world works. $y = f(x) + e$, where e is the noise and the expected value of e is 0. This describes the expected relationship between the input and output.

2.1.2 Hypothesis Class

For a linear model, $f(x) = w_0 + w_1 x$. This can be expressed in vector notation, where the weight vector $w = [w_0, w_1]$, the feature extractor $h(x) = [1, x]$. Then $f_w = wh(x) = w^T h(x) = h(x)^T w$. Then the hypothesis class is:

$F = \{f_w : w \in R^2\}$

Oftentimes, feature 1, $h_0(x)$ is often 1 (constant). Even for polynomial regression, it is still linear in the params w.

2.1.3 Loss Cost Function

Assume Gaussian Noise Model, with mean 0 and is independent. Loss Cost Function: Squaring differences of the predicted and the actual data to make it positive, and easily differentiable. It also has the effect of making large errors much more prominent.

The training loss $TrainLoss(w) = \frac{1}{|D_{train}|} \sum_{(x,y) \in D_{train}} Loss(x, y, w)$

2.1.4 Maximum Likelihood Estimation

$y^i = f(x^i) + e^i$ The target output is generated by the true function and some noise. The function $f(x^i) = \theta^T x^i$, and is linear with parameter theta. THe noise e follows a gaussian distribution with mean as 0, the Guassian Noise Assumption. And finally:

$y|x; \theta \ N(\theta^T x, \sigma^2)$ implies that the output y, given a value of x, also follows a normal distribution with $\theta^T x$ as its mean and the same varaince.

Then, the likelihood function is the product of all conditional probabilities of observing each data point given theta, assuming that they are all independent and identically distributed.

$L(\theta) = \prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}; \theta)$

$L(\theta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$

Then, we simply need to optimize the function (minimize or maximize it), and it is only dependent on theta.

Take the log of the Likelihood function to obtain:

$l(\theta) = n \log\left(\frac{1}{\sqrt{2\pi\sigma}}\right) - \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{i=1}^{n} \left(y^{(i)} - \theta^T x^{(i)}\right)^2$

Since the first part are all Constants, only the final term needs to be optimized, and this is simply the squared loss. Also note that the log function can be used since log is strictly increasing, and optimization still works.

2.1.5 Residual Sum of Squares

RSS can be converted to Matrix Form:

- $Y = Hw + e$
- $Y : n \times 1$
- $H : n \times d$
- $w : d \times 1$
- $e : n \times 1$

2.2 Gradient Descent

On a convex function, we want to find the global minimum, and on a concave, we want to find the global maximum. Functions may have multiple local minima and maxima, and the algorithm may get stuck at these optima without finding the global minimum or maximum.

- Initialize the parameters of w (the weights) to 0 or some small value $w = [0, \dots, 0]$
- Run this loop for T epochs (iterations), and in each loop perform updates to the weights
- $w = w - \eta \nabla\_wTrainLoss(w)$, where etais the step size (learning rate) and nabla is the gradient of the training loss function, which points in the direction of the steepest ascent. Adding it moves towards a maximum, and subtracting it moves towards a minimum
- A larger step size leads to faster convergence, while a small one allows for finer adjusments

2.2.1 Step Size

If it is too large, it will skip over the optima. If it is too small, it will take too long to reach the optima. A common strategy is to use a shrinking step size, such as $\eta(t) = \frac{a}{t}$ or $\eta(t) = \frac{a}{\sqrt{t}}$

2.2.2 Convergence

For convex functions, convergence happens when gradient = 0. In practice, this usually stops when the absolute value of the gradient is lower than a threshold, or when the maximum number of iterations is done.

After learning/training, a final weight for the parameters w will be obtained, and we can then feed the model inputs to get a prediction.

2.3 Performance Evaluation

For loss cost functions, apart from the squared loss, an absolute error can also be taken.

2.3.1 Perfect Predictions

Perfect Predictions implies a loss of 0. A polynomial with a large degree may be able to approximate a data set very well. However

- Too large polynomial for generalize
- Overfitting
- Only if this polynomial is the universal truth, then by all means use it

2.3.2 Training Loss

Generally, as model complexity increases, training error decreases. This may not be a priority, and small training error does not necessarily imply good predictions (UNLESS the training data includes all data you will ever see)

2.4 Generalization Error

Ideally, estimate loss over all possible data (the true error). However, not all pairs are equally likely. Ideally, weigh the losses by how likely they happen, and try to minimize the loss for those that occur more often.

For example, by overfitting, the generalization error (with respect to the one true distribution) can increase more and more as model complexity increases.

Note that it is impossible to compute the generalization error (unless somehow the entire data and distribution is available).

2.5 Test Error

Never train on things in your test set. Since it is impossible to compute generalization error, approximate it by looking at data not in the training set. Then compare it to the test set taking that as a proxy for the true data set.

Split data to hold both test and training data. The training data should be more than the test data (70/30, 80/20), though it may be 20 80?

2.6 Bias Variance

Sources of Errors: Noise, Bias, and Variance. The expected prediction error is the noise, the square of the bias, and the variance.

THere is an optimal tradeoff

2.6.1 Noise

Data is inherently noisy, and it is am irreducible error.

2.6.2 Bias

There is some true function that perfectly predicts. The bias is the difference between the perfect prediction and the actual prediction. The Bias should be as close to 0 as possible.

Bias is high when the hypothesis class is unable (not complex enough to capture the "true" function.

2.6.3 Variance

How much do specific fits from among N fits vary from the average fit (Not against the true fit). In low complexity, models typically do no vary and thus have low variance. As complexity increases, there are more specific fits possible resulting in much more variance.

As the number of data approaches infinity, training a model many times to make many different fitters will result in less variance.

3 Evaluation, Overfitting, and Bias-Variance Tradeoff

3.1 Overfitting

Has low bias, high variance, and poor generalization. It is harder to overfit a large number of observation as complexity generally increases as observations increase. An empircally observed phenomenon is a large magnitude in the estimated parameters (compared to everything else). The norm (length) of the vector will also become very large

3.1.1 Regularization

A possible fix is to incorporate the norm into the cost, and penalize the model. With regularization, cost = Measure of Fit + Measure of the Magnitude of Coefficients. This will only apply on the training (learning) set, while the actual and validation sets should only have the error = measure of fit.

Sum of Absolute: Lasso L1

Sum of Squares Ridge L2

3.1.2 Ridge

Here, let the total cost = measure of fit and the measure of magnitude of coefficients $RSS(w) + ||w||_2^2$

4

4.1 Learning Features

Before this, someone picks the features needed for us.

4.2 Neural Networks

4.3 Graph Representation

4.4 Convolution

4.5 Regularization