Notes on Pattern Recognition

Andres Ponce

March 17, 2020

Pattern Recognition is a subset of Machine Learning and Aritficial Intelligence in general. It's concerned with analyzing data, extracting particular features or patterns from that data. We will discuss severl methods for feature extraction, all the way to neural networks and just touch on deep learning!

Introduction

Here, I am using the `tufte-latex` package for LaTeX. I thought the design looked quite nice and wanted to give it a shot!

Curve Fitting

THERE EXIST two main types of problems in the field of pattern recognition. First, there is ***regression***, and there is ***categorization***. [1]

The terms *Artificial Intelligence, Machine Learning*, and *Pattern Recognition* all share common properties. PR $\land$ ML $\subset$ AI. ML atempts to make computers take in empirical data and make decisions. AI, more broadly, tries to make computers perform actions that were usually thought to be exclusive to humans.

Some different types of PR problems involve **supervised** and **unsupervised** learning. [2] There might be a couple of problems that better suit un-supervised learning, such as:

- **Clustering**: Clustering involves finding patterns in the data which more often resemble themselves rather than other data, i.e. finding similar patterns within the data.
- **Density Estimation**: Here, we try and find the probability density function given a set of points.[3]
- **Dimensionality Reduction**: When multiple variables are involved in a problem, the space in which their solution exists rises exponentially for each new variable.[4]

Although we can imagine supervised scenarios, IRL our model would probably have to find the solutions on its own.

We can imagine having a network actually *generate* new data based on the patterns its seen before. For example, given many human faces, we could create a model that generates human faces which do

[1] **regression** problems deal with the mapping of an input vector to a continuous space, whereas **categorization** takes an input and places it in a finite and discrete set of different categories.

[2] Their difference is whether their target data is available when the problem begins, i.e. whether we know the correct answer.

[3] Remember the PDF only represents a probability density over a continuous interval.

[4] i.e. solving for three variables is exp. harder than solving for two.

not exist in reality. This would be the idea of a **Generative Adversarial Network**(GAN)[5]

Finallly, curve fitting! So, we have a polynomial function, whose generic form is

$$y(x, w) = w_0 + w_1 x + w_2 x^2 + ... + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

M gives us the order of the polynomial, and if I remember my algebra correctly, it should have $M - 1$ curves. If M is too large, we *overfit* and our function will have too many curves and not match the actual function all that well.

w gives us the value of the constant mulitplier, which can be found by minimizing the **expectation** $E(w)$.[6] This expectation is fiven by

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, w) - t_n\}^2$$

Basically we want the difference between the value that our function gave us and the acutal value of the function.

Because we want to avoid overfitting, we introduce a weight term to our cost function. In Figure 1, our model tries to overcompensate because it tries to fit to every single data point, with all its discrepancies included. Therefore, if we add a specific value to our cost function, it would encourage the model to choose a curve that works within the defined parameters and is not too complex. Our new curve might be something like:

$$\tilde{E}(w) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, w) - t_n\}^2 + \frac{\lambda}{2} \|w\|^2$$

where $\|w\|^2 = w^T w = \sum_{n=0}^{M} \omega_n^2$. We could also include a linear term for w, but with its quadratic version it would be a **ridge regression**. We will have to choose our w such that $\tilde{E}(w)$ is minimized, thus if we add a bigger term at the end(quadratic), the value rises quicker and we'll choose a smaller w.

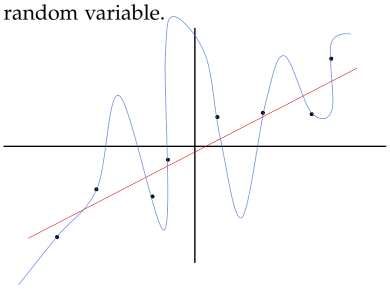

Figure 1: Overfitting leads to overly complex models. The blue line is our model.

Probability

We need to review the fundamentals of probability before moving.

The **expectation** of a probability can be thought of as the average value of a probability distribution, or the value that we should expect should we pick a random value from the distribution; the value that the distribution revolves around. The main idea is that we multiply all the possible values of $f(x)$ by the probability that it is such a way. Thus, for a discrete distribution, the expectation is

$$\mathbb{E}[f] = \sum_{x} p(x) f(x)$$

[5] A GAN could have *two* networks, one that generates the data and the other one that checks its validity.

[6] The expectation is like the average, the value we should expect to see from a random variable.

We replace the sum with an integral for a continuous distribution.

The **variance** is the expected value that a random variable deviates from the expected value. [7]

$$var[f] = \mathbb{E}[(f(x) - \mathbb{E}[f(x)])^2]$$

Covariance refers to the amount that two random variables vary together. The **Gaussian Distribution** refers to a very complicated formula that describes a probability distribution. There can be multivariate distributions that utilize a Gaussian distribution, which has a **covariance matrix** for all the variables and how they relate to each other.

Bayes's Theorem

This theorem relates the inverse conditional probability $p(w|D)$ with $p(D|w)$.

$$p(w|D) = \frac{p(D|w)p(w)}{p(D)}$$

Given the set of observations D, we can formulate how likely it was for a certain coefficient w to arise both before observing the data and after. [8]

Probability Distributions

Suppose we have an unfair coin that we are throwing in the air n times. We would like to measure the probability of getting x heads. The probability distribution would be: $p(x) = (p(x))^x(1 - p(x))^{1-n}$. This is known as the **Bernoulli Distribution**

Similar to the Bernoulli distribution, we might consider the **Binomial Distribution**, where factor in the *different* possible ways to draw M heads. The formula then becomes:

$$\text{Bin}(m|N,\mu) = \binom{N}{M}\mu^m(1-\mu)^{N-m}$$

[9]

The Beta Distribution

First, we have to talk about **prior distributions** and **posterior distributions**. These are $p(\theta)$ and $p(\theta|x)$, respectively.[10]

Together, these inform us as to how the probabilities for a random event and its complement depend mutually on each other. The hyperparameters a and b can be considtered the amount of times that the observations for $x = 1$ and $x = 0$ have been observed.

[7] **Variance** is the expected amount that the expected amount differs from $f(x)$. The value is squared so the expectation removes positive.

[8] The **prior probability** $p(w)$ of the parameter w is its likelihood before observing D. $p(D|w)$, the **likelihood function**, relates how likely the observations are given the parameter. It's gotten after observing D.

[9] This formula relates the probability fo the *combinations*, or possible orderings, in which M events happen.

[10] The difference between the two is knowing the result of some variable x which can influence the result. Together they are known as **conjugate distribution**.

This was in the textbook, not in the presentation

Regression

If we remember, when doing regression, we have a **basis function**.[11] Whatever approximation we wish to do would then probably be some sort of linear combination of the original basis function.

The basis function is of the form(we will describe the forms of ϕ below):

$$y = w_0 + w_1\phi_1(x) + ... + w_{M-1}\phi_{M-1}(x)$$

However, as with overfitting, the basis function will not necessarily allow you to automatically reach all the space at first. However, since linear functions will just allow a straight line, then we cannot use it always. Some examples of basis functions include:

The key to understand is the basis function ϕ. Because we have an input vector $\{X\}$ with n elements, each of the x_n will have to be passed through ϕ.

- **Polynomial**: These functions are of the form:

$$\phi_j(x) = x^j$$

- **Gaussian**: governed by μ_j and s.[12]

$$\phi_j(x) = \exp\{-\frac{(x-\mu_j)^2}{2s^2}\}$$

- **Sigmoidal**:

$$\phi_j(x) = \phi(\frac{x-\mu_j}{s}) \text{ where } \phi(a) = \frac{1}{1+\exp(-a)}$$

In the figure below, t refers to the **target vector**, which is the result of the actual function $y(x, w)$ and a random gaussian nose ϵ [13]

$$p(t|x, w, \beta) = \prod_{n=1}^{N} \aleph(t_n|w^T\phi(x_n), \beta^{-1})$$

Often times, to find the maximum of the likelihood function, we instead take its natural logarithm. We do this because the function itself is **monotonically increasing**, meaning that maximizing the likelihood function is equivalent to maximizing the **log likelihood function**.

[11] A **basis function** is a function that allows us to reach any point within a given space. I think it might be similar to how you can reach any point in 3D with three orthogonal vectors?

[12] Notice how μ_j will determine how big our numerator and denominator are, i.e. the expected value tells us more than anything what to expect.

[13] Essentially, this formula says that the likelihood of a given target vector t given a certain input vector x, parameter w, and precision β is the product of the normal distribution of each member in t. This makes sense since the probability that our target vector is a certain way would depend on the normal distribution. And since every x is independently and identically distributed, having the product of all of them should give the correct overall likelihood. Right? :)

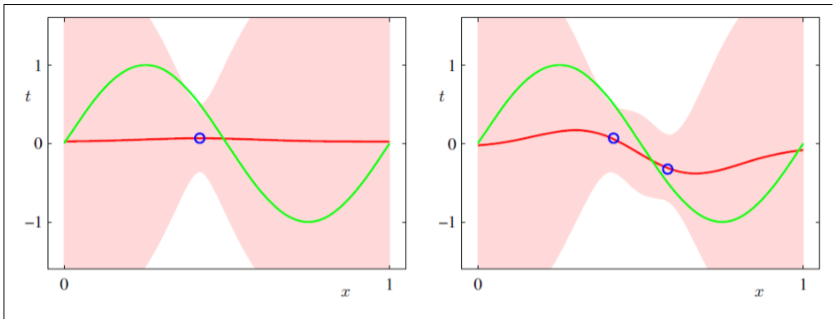

Figure 2: The red line is our predictive function; the green is our actual function which we don't know, and the red area reps. 1 standard deviation from the function value at each point.

Least Squares

The idea behind a **Least Squares Approximation** is to take the difference between the function value and the one our model gives us. We take this difference (squared to maintain a positive number) and try to minimize over the whole sum of errors. Then, optimality would involve minimizing the log likelihood function.

However, least squares solutions can also suffer from **overfitting**, so we can add a **regulartization**. This term ensures that our approach does not overfit the actual answer again. As metnitoned, we can add a linear term or a quadratic term. The general form for a good equalizer might be:[14]

$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - w^T\phi(x_n)\}^2 + \frac{\lambda}{2}\sum_{j=1} M|w_j|^q$$

Multiple Outputs

Suppose we have target vector with multiple components. The way to find the best coefficients would be to have either multiple basis functions or to estimate the components of **t**. This would be equivalent to K indpendent regresion problems for K dimensions.

Maximum a posterior

In regression problems, both the prior and posterior probabilities are Gaussian. Thus[15]

$$p(w) = \aleph(w, m_0, S_0)$$

and

$$p(w|\mathbf{t}) = \aleph(w|m_N, S_N)$$

are both Gaussian in nature.

After that, we finally start using the predictive function $y(x, w)$ to guess the points.[16]

[14] where t is target vector,w would be the parameter matrix? ϕ is the basis function

[15] m_0 and S_0 are the mean and covariance matrix respectively.

[16] Remember this function takes the parameters and multiplies them by the basis function, and adds some Gaussian noise just cuz.