CMPT 419/726 — Machine Learning

Salehen Shovon Rahman

September 17, 2015

Personal notes regarding machine learning.

Polynomial Curve Fitting

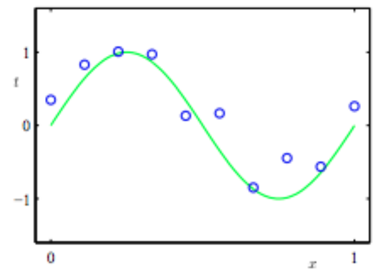


Figure 1: An example data set where we want to fit a polynomial curve into.

Here's an example machine learning problem: try to find the best polynomial that can potentially fit a set of data points, and have it be fit as best as possible. This is known as the polynomial curve fitting problem, and it's a supervised regression learning problem.

The Problem

Suppose we are given a training set of N observations, $(x_1, x_2, \ldots, x_N)$ and $(t_1, t_2, \ldots, x_N)$, $x_i, t_i \in \mathbb{R}$. We want to find a polynomial $y(x)$ that fits these data the best.

Let's start out by defining a $y(x, \mathbf{w})$.

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{i=1}^{M} (w_i x^i) \tag{1}$$

How do we measure success? Or, a better question, for what values of the coefficients $\mathbf{w}$ will yield the best results? To answer that, we define an error function E.

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{N} (y(x_i, \mathbf{w}) - t_i)^2 \tag{2}$$

We then use the $\arg\min_x f(x)$ function to find the value for the parameter that yields the lowest value in a given function.

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} E(\mathbf{w}) \tag{3}$$

$\min_x f(x)$ finds the lowest possible value for the expression $f(x)$, while $\arg\min_x f(x)$ finds the value for x where $f(x)$ would be the lowest.

So, in other words, we want to find a $\mathbf{w}$ such that $E(\mathbf{w})$ is the lowest among the set of all possible values of $\mathbf{w}$.

Except, the attempt at finding a value $\mathbf{w}^*$ such that $E(\mathbf{w}^*) = 0$ can become problematic.

Earlier, we mentioned that we had an initial set of training data. However, for most cases, when trying to fit the polynomial such that $E(\mathbf{w}^*) = 0$, for the training set, we risk having it so that when a test data set is introduced, the error function yields a high value. This is known as *overfitting*.

In the end of the day, although we want the curve to fit the data as best as possible, we also want a *genralization* derived from the given training.

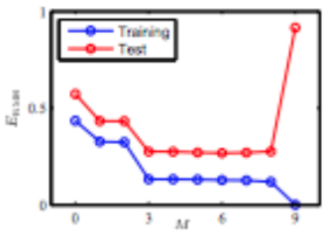


Figure 2: As we can see, the first few polynomials of degree $N < 9$ fit the data fine, even when test data is introduced to the training set, but misses the mark entirely when $N = 9$. This is the result of overfitting.

BUT FIRST, before we go ahead with finding a good gneralization, for convenience, instead of just using the error function E, we use the root-mean-square (RMS) error function, defined by

$$E_{\text{RMS}} = \sqrt{2E(\mathbf{w}^*)/N} \tag{4}$$

According to the text book, *Pattern Recognition and Machine Learning*, the reason why we are using RMS, is the following:

> [The] division by N allows us to compare different sizes of data sets on an equal footing, and the square root ensures that E_{RMS} is measured on the same scale (and in the same units) as the target variable t. (p. 7)

Now, to actually tune our $\mathbf{w}$ for better generalization, we can split our training data into two sets: training set and validation set. In the case of finding the polynomial, the training set can be used to find each $w_i \in \mathbf{w}$, and the validation set is used to optimize the complexity, which can be represented by M (the size of $\mathbf{w}$), or a λ, which will be discussed later.

There are several techniques used to control overfitting.

THE FIRST TECHNIQUE to avoid overfitting is cross-validation.

Here, we group the data into separate sets. We first "train" our parameters to a union of all the separated set, while leaving one out. Then we optimize by including the one we initially excluded. Afterwards, we "train" again with a new union of our sets, while leaving yet another one out, but including the one that we initially left out, all the way until no sets are left to "leave out".

AND THEN, there's regularization for controlling over-fitting.

Notice how the oscillation increases as M increases? This is because the magnitudes of the coefficients in $\mathbf{w}$ increases as M increases.

In order to avoid high coefficient magnitudes, we can "penalize" them using a modified error function, by adding a $\frac{\lambda}{2}\|\mathbf{w}\|$ term to the

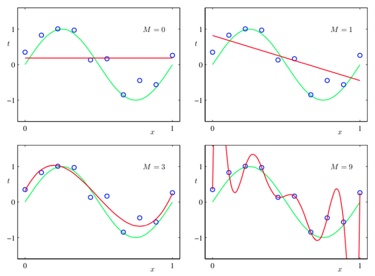


Figure 3: Visually, we see that the oscillation increases as M increases.

	$M=0$	$M=1$	$M=6$	$M=9$
w_0	0.19	0.82	0.31	0.35
w_1		-1.27	7.99	232.37
w_2			-25.43	-5321.83
w_3			17.37	48568.31
w_4				-231639.30
w_5				640042.26
w_6				-1061800.52
w_7				1042400.18
w_8				-557682.99
w_9				125201.43

Table 1: For higher values of M, we see the magnitude of w_i increasing

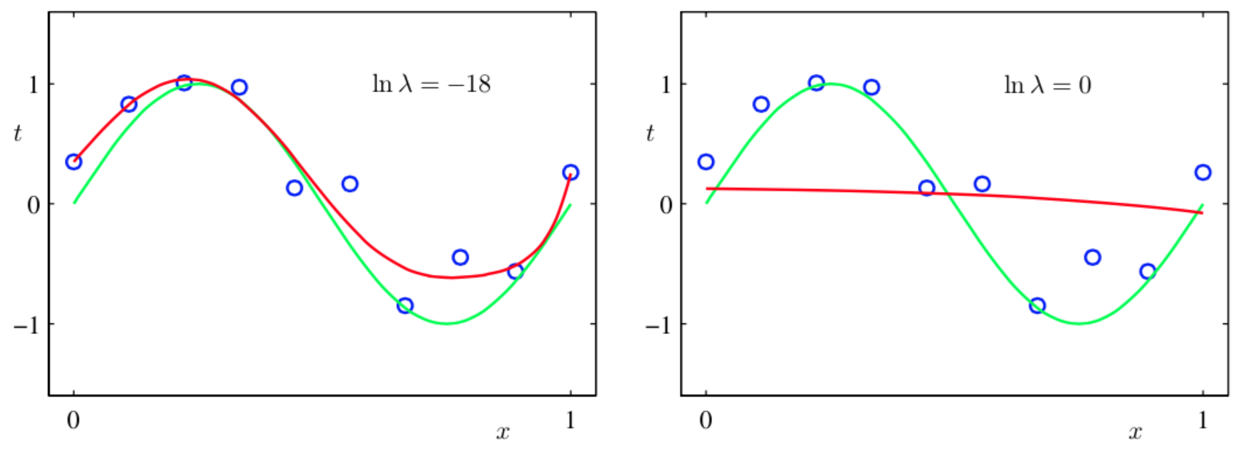
original error function.

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(y(x_i,\mathbf{w}) - t_i)^2 + \frac{\lambda}{2}\|\mathbf{w}\| = E(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\| \tag{5}$$

If we are to now apply the error function to our trials, we see that the ceofficient are no longer large.

	$\ln\lambda = -\infty$	$\ln\lambda = -18$	$\ln\lambda = 9$
w_0	0.35	0.35	0.13
w_1	232.37	4.74	-0.05
w_2	-5321.83	-0.77	-0.06
w_3	48568.31	-31.93	-0.05
w_4	-231639.30	-3.89	-0.03
w_5	640042.26	55.28	-0.02
w_6	-1061800.52	41.32	-0.01
w_7	1042400.18	-45.95	-0.00
w_8	-557682.99	-91.53	0.00
w_9	125201.43	72.68	0.01

Table 2: For higher values of λ, the coefficient magnitudes are much lower, possibly even near 0



As you can see here, even for $M = 9$, previously, the polynomial curve deviated wildly, which could have potentially yielded high error values relative to new potential data

FINALLY, there's the third option: just get more data! A rule of

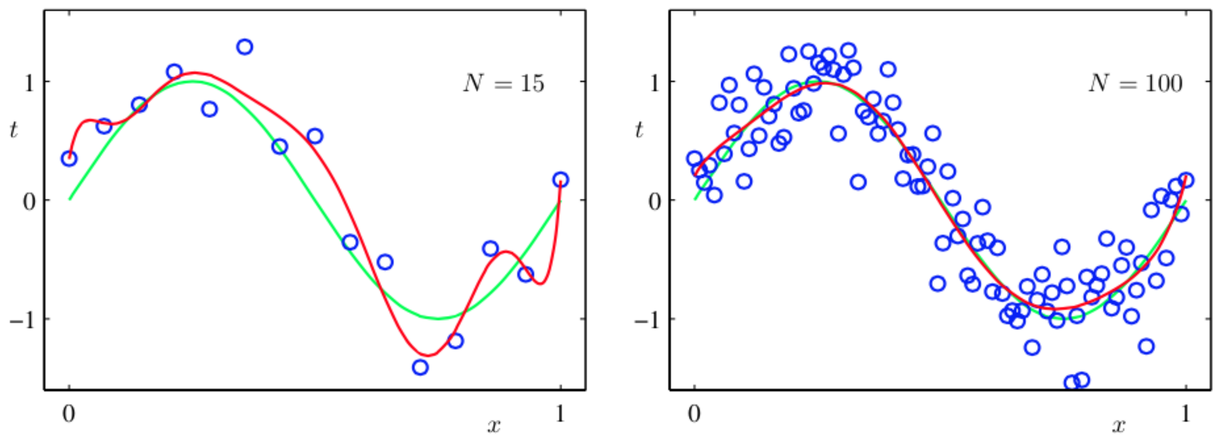


Figure 4: On the left, we see an *attempt* at having the curve fitted to the data we have so far; on the right, we see more data added, and the curve appears show a much better approximation.

thumb is that the number of datapoints should not be any less than five times the number of adaptive parameters.

Probability Theory

Random Variables

THE TERM "random variable" is can be misleading. A random variable is not a variable at all. Instead, it's a mapping from a *random event* to a possible outcome.

So here's an example random variable:

$$X = \begin{cases} 1 \text{ if heads} \\ 0 \text{ if tails} \end{cases} \tag{6}$$

As we can see here, X is a mapping from the set $\{\text{heads}, \text{tails}\}$, to the set $\{1, 0\}$.

More informally, a random variable is a "label" to a given element of a set of possible events.

To add to the confusion, when we say that we have an event from our random variable (for example, an event from X defined above), we express it as $X = 0$ or $X = 1$, etc.

The end purpose of random variables is to derive a probability distribution given a value $x \in \{\text{Range of X}\}$. So, going back to the above head/tail example, if we wanted to express the probability of getting heads, we write $p(X = 1)$, likewise, for expressing the probability of getting tails, we write $p(X = 0)$, and so and so forth.

The purpose of using random variables is for us to easily be able to write inequalities inside our probability expression. So, for instance, let's define a random variable Y that contains a mapping of a list of combinations of two dices to the sum of their faces. We can easily express the probability of a roll, for example, a value greater than 10, like so: $p(Y > 10)$.