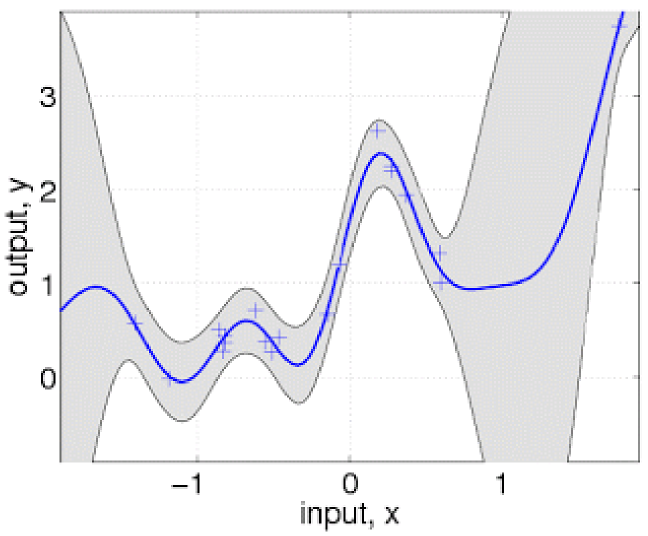
10th May 2016

A Simple Intro to Gaussian Processes (a great data modelling tool)

Gaussian Processes (GPs) are a powerful technique for modelling and predicting numerical data. Being both relatively new and mathematically quite complex, they're not as well known as other techniques. They have some strong advantages:

1. Flexible: Can be used to model many different patterns.
2. You make fairly few assumptions about the model.
3. Based on probability theory, so they have a solid mathematical grounding.

This article is an easy-to-read introduction. Instead of diving into the maths behind a Gaussian Process, let's start with a simpler algorithm.



A Gaussian Process, fitted using MatLab, showing most-likely-value & confidence interval. Note how the shape hugs the data, and how the uncertainty varies depending on the data - sometimes the model is confident, sometimes it isn't, and you know which is which.

K Nearest Neighbours (KNN)

K Nearest Neighbours is a classic AI algorithm, and very easy to understand. It also provides a surprisingly good basis for explaining Gaussian Processes (without any maths involved yet). Here's how KNN works:

The task: Given an item *x*, predict the category of *x*. E.g. you might ask, Is this email spam or not-spam?
The method:

1. Let's pick k=5.
2. Given an input item *x*... E.g. a fresh email has arrived, is it spam?
3. Look through your training data, and find the 5 items most similar to the input item *x*. These are the nearest neighbours.
4. Look at the categories those 5 items have, and predict the most common category from them.
5. Done :)

A few things to note about the KNN algorithm:

- It is a lazy algorithm. The model isn't trained in advance, as with say Linear Regression or Naive Bayes -- instead the work is done when an input item is presented.
- The data *is* the model. If you have enough data, KNN can model a really wide range of patterns. Rather than being forced into a particular shape, the data can speak for itself. There is a cost to this though - it requires more training data.
- The key part is judging when items are similar. How you do that will depend on the problem you're looking at.

These are also key properties of Gaussian Processes.

The classic KNN algorithm is for predicting categories (e.g. spam / not-spam), but we can modify it as follows to make numerical predictions (e.g. the price of fish):

Step 4': Having found the k nearest neighbours, take the average value.

From KNN to a Gaussian Process

So what is a Gaussian Process?
It deals in numerical data, e.g. the price of fish -- and for the rest of this article, let's say it *is* the price of fish as a function of date that we're modelling. We keep all the training data, and say that when predicting new data points from old ones, the uncertainty/noise will follow a multivariate Gaussian distribution. The relationship given by that distribution lets us make predictions from the training examples.

For non-Mathematicians, let me briefly cover the terminology from that paragraph. A distribution tells you how likely different values are. The Gaussian Distribution, aka the Normal Distribution, has a bell-shaped curve: the mean is the most likely point, and the probability drops off rapidly as you move away from the mean. The multivariate Gaussian (which is what we want) can specify correlations between multiple variables. The Gaussian Distribution naturally arises in lots of places, and is the default noise model in a lot of machine learning.

Similar to KNN, the Gaussian Process is a lazy algorithm: we keep the training data, and fit a model for a specific input. Also like KNN, the shape of the model will come from the data. And as with KNN, the key part is the relationship between examples, which we haven't defined yet...

Introducing the Kernel Function

A Gaussian Process assumes that the covariance between any set of points is a multivariate Gaussian.
The "variables" here are the different values for the input *x*. In our price-of-fish example, the input is the date, and so every date gives a dimension! Yes, in principle, there are an infinite number of variables! However we only have a limited set of training data -- which gives us a finite set of "variables" in the multivariate Gaussian -- one for each training example, plus one for the input we're trying to predict.

A multivariate Gaussian is defined by it's mean and covariance matrix, so these are the key things for us to calculate.

The kernel function of a GP says how similar two items are, and it is the core of a specific Gaussian Process. There are lots of possible kernel functions -- the data analyst (e.g. you) picks an appropriate one.

The kernel function takes in any two points, and outputs the covariance between them. That determines how strongly linked (correlated) these two points are.

A common choice is to have the covariance decrease as the distance between the points grows, so that the prediction is largely based on the near neighbours. E.g. the price of fish today is more closely linked with the price yesterday than the price last month. Alternatively, the kernel function could include a periodic part, such as a sine wave, to model e.g. seasonal ups and downs. The Wikipedia article lists some example kernel-function formulas.[1]

The kernel function will often have some parameters -- for example, a length parameter that determines *how* quickly the covariance decreases with distance. These parameters are found using optimisation software -- we want parameters that optimise the likelihood of the observed data. We can write down the probability of the observed data (the likelihood) as a function of the kernel function parameters, and then pick kernel function parameter values to maximise the likelihood.

Given an input *x* (e.g. "next Thursday") with the training examples *x*1, *x*2, ... *xn* (e.g. the price of fish each day for the last few months), then the GP model is that (x, x1, x2, ...*xn*) has a distribution with mean 0 and a covariance matrix defined by the kernel function.

From the covariance matrix, you can then calculate the prediction for *x*. The prediction for *x* (or in probability theory terminology, the marginal distribution for *x)* will be a simple one-dimensional Gaussian. It has a mean value (which is the most likely value for *x*) and a standard-deviation for the uncertainty.

Building a GP

This article has skipped over the technical details of how you carry out certain steps. I've blithely written about "optimising the likelihood" without saying how you do that. That's partly because there are multiple ways, and partly to keep this article simple. The short answer is: you'll be using software of course, and most likely software that someone has kindly already written for you.

I'm not going to recommend a particular software tool here, as the choice really depends on what you're familiar with and where you're using it. There are GP calculators available for many environments, e.g. Weka has one for Java [2], or you can talk to your local Winterwell office[3] :)

Going Deeper

[1] Commonly used kernel functions, in Wikipedia: https://en.wikipedia.org/wiki/Gaussian_process#Usual_covariance_functions [https://en.wikipedia.org/wiki/Gaussian_process#Usual_covariance_functions]
[2] Weka [http://www.cs.waikato.ac.nz/ml/weka/] , a Java toolkit for machine learning.
[3] Winterwell, data science consultancy [http://www.winterwell.com/]

So you want to know the mathematical details? Good for you!
Try reading these resources:

[4] "Gaussian Processes: A Quick Introduction" [http://arxiv.org/find/math,stat/1/au:+Ebden_M/0/1/0/all/0/1] by Mark Ebden.
[5] "Gaussian Processes for Machine Learning" [http://www.gaussianprocess.org/gpml/chapters/] by Carl Rasmussen and (my boss once-upon-a-time) Chris Williams.

Posted 10th May 2016 by Daniel Winterstein

0 Add a comment