Summary of Gaussian process classification (GPC)

Alexander

July 16, 2017

A note on the syntax used in the equations in here: vectors and matrices are denoted with bold font $\mathbf{x}$ and the scalar equivalent uses normal font x. Subscripts on scalars x_n generally indicate its index within a vector or matrix, and subscripts on vectors $\mathbf{x}_n$ indicates the length of the vector.

1 Background stuff and definitions

- *Gaussian process (GP)*: Generalization of the Gaussian probability distribution to multi-dimensional space. According to wikipedia, the Gaussian process is characterized by the joint probability of all the random variables/dimensions of a continuous space (in essence a multivariate random distribution). According to [1], a Gaussian process governs the properties of functions, this is clarified further in [2] which explains that if the weights to a set of basis functions have a Gaussian distribution, then sampled weights will result in random functions of a Gaussian process.
- *Joint probability*: $p(y, \mathbf{x})$ the probability of y, a class label and $\mathbf{x}$, a data point
- *Posterior probability*: $p(y|\mathbf{x})$ probability of y given $\mathbf{x}$, probability of x after x has been selected
- *Prior probability*: the probability distribution you believe describes the process before new data is taken into account (often assumed to be Gaussian)
- *Latent function*: Hidden function describing the probability of each class over a given space.
- *Bayes theorem* says joint probability $p(y, \mathbf{x}) = p(y)p(\mathbf{x}|y) = p(\mathbf{x})p(y|\mathbf{x})$
  - therefore $p(y|\mathbf{x}) = \frac{p(y)p(\mathbf{x}|y)}{p(\mathbf{x})}$
- The definition of Bayes theorem gives rise to two modeling approaches: generative and discriminative:
  - The generative approach models the class conditional distributions $p(\mathbf{x}|y)$ for each class (typically done by fitting Gaussians: $p(\mathbf{x}|C_c) =$

$\mathcal{N}(\mu_c, \Sigma_c)$ to the existing data) to apply Bayes theroem. Note that this assumes that each class has a Gaussian distribution. Also, density estimation (Σ_c) is difficult for high dimensional $\mathbf{x}$.

- The discriminative approach tries to model the posterior ($p(y|\mathbf{x})$) directly where any number of approaches are valid as long as they produce probabilistic results in the range $[0, 1]$.

- The classifiers developed in the book [1] use the discriminative approach

2 Kernel methods

GP regression and classification are kernel methods, therefore it is necessary to identify what that means. Here I will try to explain it following the kernel methods chapter in [2]. Kernel models are similar to linear parametric models in that they are a re-casting of the problem, this is called the duel representation. The kernel method is based on a fixed nonlinear feature space mapping (or basis function) $\phi(\mathbf{x})$ which can be any function. The kernel function is given by

$$k(\mathbf{x}, \mathbf{x}^{'}) = \phi(\mathbf{x})^T \phi(\mathbf{x}^{'}) \tag{1}$$

where $\mathbf{x}$ and $\mathbf{x}^{'}$ are points in the sample set. By computing all kernels, we end up with the Gram matrix $\mathbf{K}$, an $N \times N$ symmetric matrix where each element contains the kernel function

$$K_{nm} = \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m). \tag{2}$$

Because the kernel is a dual representation of linear parametric methods (which are easy to understand), we can show that it has an error function analogous to the regularized sum-of-squares error function. Under the linear model, where we solve for weights, the error function looks like

$$J(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} (\mathbf{w}^T \phi(\mathbf{x}_n) - t_n)^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w} \tag{3}$$

where $\mathbf{w}$ are the weights, N is the number of training samples, t_n is target or label n, and λ is the weight regularization constant (this puts pressure on the weights so that they don't get too large, which guards against over-fitting). Without going into all of the math, the kernel method can represent the error function in terms of the Gram matrix and a new vector $\mathbf{a}$, defined by $\mathbf{a}_n = -\frac{1}{\lambda}(\mathbf{w}^T)\phi(\mathbf{x}_n - t_n)$.

$$J(\mathbf{a}) = \frac{1}{2} \mathbf{a}^T \mathbf{K}\mathbf{K}\mathbf{a} - \mathbf{a}^T \mathbf{K}\mathbf{t} + \frac{1}{2} \mathbf{t}^T \mathbf{t} + \frac{\lambda}{2} \mathbf{a}^T \mathbf{K}\mathbf{a} \tag{4}$$

is equivalent to equation 3. We want to minimize the loss over our vector $\mathbf{a}$ by setting the gradient of $J(\mathbf{a})$ to zero. Doing this, we get

$$a = (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{t} \tag{5}$$

and substituting $\mathbf{a}$ into the linear regression model, we can predict $y(\mathbf{x})$ given any new input $\mathbf{x}$

$$y(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) = \mathbf{k}(\mathbf{x})^T(\mathbf{K} + \lambda\mathbf{I}_N)^{-1}\mathbf{t} \tag{6}$$

where $\mathbf{k}(\mathbf{x})$ is a vector with elements $k_n(\mathbf{x}) = k(\mathbf{x}_n, \mathbf{x})$. So we see that through building a kernel function and the Gram matrix, we can derive a loss function and optimize it's parameters, which in turn lets us predict the output $y(\mathbf{x})$ of new input vectors $\mathbf{x}$. This gives us the framework for a regression model.

3 Gaussian processes for classification

GPs are an extension of kernel methods to probabilistic discriminative models. Kernels arise naturally in a Bayesian setting, so things get re-derived using probability. Using GP, we model the space of probable solutions rather than the parametric solutions themselves. We will see that the Gram matrix can give us the covariance of the probability density function of the solution space. In essence GP is a way to abstract one layer above a individual parametric model, providing a distribution of probable models (basically modeling the models, super dope).

3.1 Broadly speaking

GP for classification builds on GP for regression, which is roughly described by the Kernel methods description above. Essentially a GP regression model is built, and its output is run through a nonlinear activation function such as the sigmoid

$$\sigma(x) = \frac{1}{1 + e^x}, \tag{7}$$

which squeezes the input into the continuous range $[0, 1]$. But before we get ahead of ourselves, I will present the derivation from [2] below.

3.2 Derivation using Bayesian statistics

From the top, we begin with training data (vectors) denoted by $\mathbf{x}_1, ..., \mathbf{x}_n$ and corresponding targets (1 or 0) $\mathbf{t} = (t_1, ..., t_N)^T$. For the purpose of this derivation, a test point $\mathbf{x}_{N+1}$ is considered with target t_{N+1}. The goal is to determine the predictive distribution $p(t_{N+1}|\mathbf{t})$.

First, a GP prior is defined to map the input $\mathbf{x}_n$ to t_n. This vector looks like $a(\mathbf{x}_1), ..., a(\mathbf{x}_{N+1})$, and is defined by the Gaussian distribution

$$p(\mathbf{a}_{N+1}) = \mathcal{N}(\mathbf{a}_{N+1}|\mathbf{0}, \mathbf{C}_{N+1}) \tag{8}$$

where $\mathbf{C}_{N+1}$ is the Gram matrix, but may be combined with a term ν to ensure it is positive semidefinite (so it can be inverted). $\mathbf{C}_{N+1}$ has elements

$$C(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) + \nu \tag{9}$$

where $k(\mathbf{x}_n, \mathbf{x}_m)$ defines a kernel function. The kernel may have hyperparameters θ (a kernel function can be any function that yields a positive semidefinite Gram matrix), but we will not discuss them or how they can be optimized, because that really complicates things!

Now assuming the need for only a binary classifier, we can confine our predictor to the probability distribution of one of the classes being true, $p(t_{N+1} = 1|\mathbf{t}_N)$. This looks like

$$p(t_{N+1}|\mathbf{t}_N) = \int p(t_{N+1} = 1|a_{N+1})p(a_{N+1}|\mathbf{t}_N)\mathrm{d}a_{N+1} \tag{10}$$

where we know $p(t_{N+1} = 1|a_{N+1}) = \sigma(a_{N+1})$, but we don't know $p(a_{N+1}|\mathbf{t}_N)$. Since we don't know this, the equation is intractable. This is where the Laplace approximation comes in, it lets us approximate the probability distribution of the Gaussian posterior of a_{N+1} using another Gaussian... which is a little crazy if you ask me.

3.3 Laplace approximation

Bayes theorem is used to derive the following approximation of the posterior distribution over a_{N+a} needed to solve the the predictive equation 10:

$$p(a_{N+1}|\mathbf{t}_N) = \int p(a_{N+1}|\mathbf{a}_N)p(\mathbf{a}_N|\mathbf{t}_N)\mathrm{d}\mathbf{a}_N. \tag{11}$$

We need two distributions to solve the integral: $p(a_{N+1}|\mathbf{a}_N)$, which is the distribution of a_{N+1} conditioned on the known values of $\mathbf{a}_N$, and the posterior $p(\mathbf{a}_N|\mathbf{t}_N)$. We can define

$$p(a_{N+1}|\mathbf{a}_N) = \mathcal{N}(a_{N+1}|\mathbf{k}^T\mathbf{C}_N^{-1}\mathbf{a}_N, c - \mathbf{k}^T\mathbf{C}_N^{-1}\mathbf{k}) \tag{12}$$

where the derivation of $\mathbf{k}^T\mathbf{C}_N^{-1}\mathbf{a}_N$ and $c - \mathbf{k}^T\mathbf{C}_N^{-1}\mathbf{k}$ can be found in [2]. The first term is the expected value of a new point $\mathbf{x}_{N+1}$ where $\mathbf{k}$ is its covariance vector calculated by the kernel function. The later term defines the variance of the new point.

Next the posterior of $\mathbf{a}_N$, or $p(\mathbf{a}_N|\mathbf{t}_N)$ is required to evaluate equation 11. This is where the Laplace approximation actually comes in. We first apply a Gaussian prior to $\mathbf{a}_N$, which might be a poor assumption at first, but this distribution gets conditioned based on the known targets $\mathbf{t}_N$ using the maximum log-likelihood function. The maximum log-likelihood is found over the vector a_N. The Taylor expansion of the logarithm of $p(\mathbf{a}_N|\mathbf{t}_n)$ simplifies to

$$\Psi(\mathbf{a}_N) = -\frac{1}{2}\mathbf{a}_N^T\mathbf{C}_N^{-1}\mathbf{a}_N - \frac{N}{2}\ln(2\pi) - \frac{1}{2}\ln|\mathbf{C}_N| + \mathbf{t}_N^T\mathbf{a}_N - \sum_{n=1}^{N}\ln(1+e^{a_n}) + const. \tag{13}$$

This function is maximized using the Newton-Raphson method which requires the second derivative (the Hessian) of $\Psi(\mathbf{a}_N)$. The optimization yields a new vector $\mathbf{a}_N^*$. At this point, the Hessian is evaluated at $\mathbf{a}_N^*$

$$\mathbf{H} = \mathbf{W}_N + \mathbf{C}_N^{-1} \tag{14}$$

where $\mathbf{W}_N$ is a diagonal matrix with elements $\sigma(a_n)(1 - \sigma(a_n))$ evaluated using vector $\mathbf{a}_N^*$. Now we can define the desired posterior distribution $p(\mathbf{a}_N|\mathbf{t}_N)$ approximated by a Gaussian:

$$q(\mathbf{a}_N) = \mathcal{N}(\mathbf{a}_N|\mathbf{a}_n^*, \mathbf{H}^{-1}) \tag{15}$$

giving us all required elements to evaluate equation 11, which yields the predictive distribution of any new sample $\mathbf{x}_{N+1}$

4 Other tidbits I've gathered

- The goal of the GPC is to model the posterior probability of the target variable (events vs non-events) and map it to the range $[0, 1]$ using a nonlinear activation function (equation 7). The posterior is gathered from the training data.
- Linear basis function models require the inversion of an $M \times M$ matrix, whereas the GP methods require the inversion of a $N \times N$ matrix, which usually is a much larger matrix (M would be the number of basis functions and N is the number of data points in your training set). This is a drawback of GP because the computation time of matrix inversion is $O(N^3)$, making it impractical to do GP modeling for large datasets. There are workarounds however, such as intelligently selecting samples from your training set for building your model (this is done in SVM and SVR).
- The advantage of GP is that it expresses the distribution of an infinite set of basis function models, or a model with an infinite number of basis functions. This means that a GP model is expressed as a distribution and inherently contains information about the uncertainty of the model with respect to the training data.
- Gaussian process regression is the same things as kriging which was developed independently by the statistics community.

References

[1] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer Science+Business Media, LLC, 2006.