BIOS 835: Support Vector Machines and Gaussian Processes

Alexander McLain

October 17, 2023

Introduction

- Similar to last time, let $\boldsymbol{X} \in \mathbb{R}^p$ and $Y \in \{-1, 1\}$ and we want to predict what class each observation comes from from $\boldsymbol{X}$.
- Today, we'll generalize the methods from the last class to the non-linear case.
- Non-linear support vector machines are what most people consider to be support vector machines.
- Linear support vector machines are usually referred to as support vector classifiers (terminology we'll use here), but that is not universal.

Support Vector Machines

- From last time, the linear support vector classifier solves

$$\min_{\beta_0,\beta} \left\{ \frac{1}{2}||\boldsymbol{\beta}||^2 + C\sum_{i=1}^{n} \xi_i \right\}$$

such that $\xi_i \geq 0$ and $y_i(\beta_0 + \boldsymbol{x}_i'\boldsymbol{\beta}) \geq 1 - \xi_i$ for all i.

- After solving the derivatives of the Lagrangian (primal) function wrt $(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha})$ we get the Lagrangian (Wolfe) dual objective function

$$F_D(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j (\boldsymbol{x}_i'\boldsymbol{x}_j)$$

with $\hat{\boldsymbol{\beta}} = \sum_{i=1}^{n} \hat{\alpha}_i y_i \boldsymbol{x}_i$

Support Vector Machines

- We could, of course, consider transformations of $\boldsymbol{x}$ such as the basis functions we discussed previously.
- For example, suppose we use $h(\boldsymbol{x}) = \{h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots, h_M(\boldsymbol{x})\}$ as the variables for our support vector classifier, and

$$\hat{f}(\boldsymbol{x}) = h(\boldsymbol{x})'\hat{\boldsymbol{\beta}} + \hat{\beta}_0$$

  is our classifier.
- Sometimes by projecting the data into higher dimensions, we can make non-separable data separable.

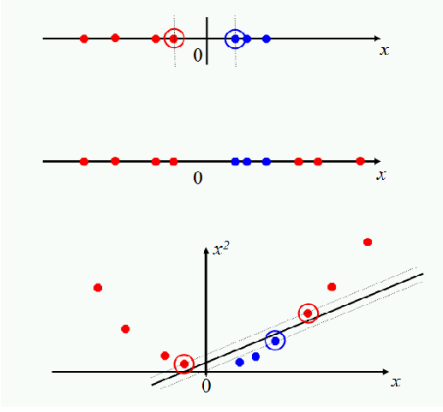
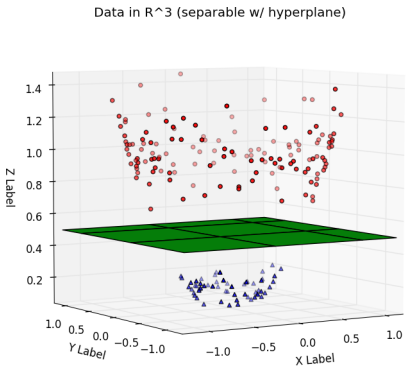


Figure: Here $\boldsymbol{x} = x$ and $h(\boldsymbol{x}) = \{x, x^2\}$.



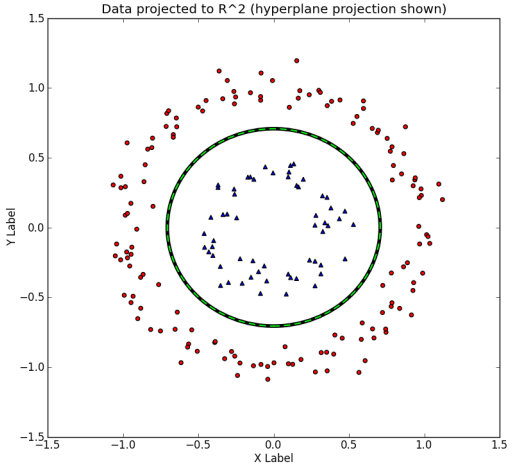


Figure: Here $\boldsymbol{x} = (x_1, x_2)$ and $h(\boldsymbol{x}) = \{x_1^2, \sqrt{2}x_1x_2, x_2^2\}$.

Support Vector Machines

- Using $h(\boldsymbol{x})$, we then have the Lagrangian (Wolfe) dual objective function

$$F_D(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle h(\boldsymbol{x}_i), h(\boldsymbol{x}_j) \rangle$$

and using the form of $\boldsymbol{\beta}$

$$\begin{aligned} f(x) &= h(\boldsymbol{x})' \boldsymbol{\beta} + \beta_0 \\ &= \sum_{i=1}^{n} \alpha_i y_i \langle h(\boldsymbol{x}), h(\boldsymbol{x}_i) \rangle + \beta_0 \end{aligned} \tag{1}$$

Systematic functional mappings

- Suppose $h \in \mathcal{H}$ is a general mapping of a p-dimensional covariate vector $\boldsymbol{x}$.
- For example, a quadratic with all two-way interactions

$$h(\boldsymbol{x}) = (x_1^2, \ldots, x_p^2, x_1x_2, x_1x_3, \ldots, x_1x_p, \ldots, x_{p-1}x_p)$$

- The dimension of h blows up!
- Calculating $\sum_i \sum_j \langle h(\boldsymbol{x}_i), h(\boldsymbol{x}_j) \rangle$ gets very expensive.
- Also, what if you wanted to use an infinite dimensional $h(\boldsymbol{x})$

Kernal Trick

- **Kernal Trick,** replace
$$\langle h(\boldsymbol{x}_i), h(\boldsymbol{x}_j)\rangle = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$
where $K(\cdot,\cdot)$ is a Kernal function.
- The choice of $K(\cdot,\cdot)$ implicitly determines h and $\mathcal{H}$.
- Given a Kernal we don't have to pick h.
- Calculating $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is much easier then calculating $\langle h(\boldsymbol{x}_i), h(\boldsymbol{x}_j)\rangle$

Examples of Kernals

- Polynomial of degree d:$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (1 + \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle)^d$
- For 2 inputs with $d = 2$, we have

$$\begin{aligned} K(\boldsymbol{x}_i, \boldsymbol{x}_j) &= (1 + \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle)^d \\ &= (1 + x_{i1}x_{j1} + x_{i2}x_{j2})^2 \\ &= 1 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} + (x_{i1}x_{j1})^2 + (x_{i2}x_{j2})^2 + 2x_{i1}x_{j1}x_{i2}x_{j2}. \end{aligned}$$

which is the same as $h(\boldsymbol{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, , \sqrt{2}x_1x_2)$

Examples of Kernals

- Radial basis: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma ||\boldsymbol{x}_i - \boldsymbol{x}_j||^2)$.
- Neural network: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(\kappa_1 \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + \kappa_2)$.
- The solution will be given by

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n} \hat{\alpha}_i y_i K(\boldsymbol{x}, \boldsymbol{x}_i) + \hat{\beta}_0$$

Considerations

- The role of the cost parameter C is clear here since perfect separation is often achievable.
- A large C will discourage any positive ξ_i, and overfit the model with a wiggly boundary.
- A small C will encourage a small value of $||\beta||$ resulting in smooth $f(x)$

Pros of SVM

- Effectiveness in high-dimensional spaces: SVMs are particularly effective in cases where the number of dimensions is greater than the number of samples.
- Versatility: Custom kernels can be used for the decision function.
- Accuracy: SVMs generally provide accurate classifiers.

Cons of SVM

- **Complexity and memory:** SVMs can be painfully inefficient to train. They use a significant amount of memory because they need to maintain an instance of each support vector in the model.
- **No direct probability estimation:** SVMs do not directly provide probability estimates; these are calculated using an expensive five-fold cross-validation.
- **Difficult (Black-box) Interpretation:** The results of predictions can be very difficult to interpret compared to some other algorithms.
- **Tuning Required:** The success of an SVM is heavily dependent on the correct tuning of the hyperparameters.
- **Poor performance with noise:** If the data is noisy or overlaps, it can lead to a decrease in performance as SVMs are sensitive to the noise in the dataset.

Gaussian Processes

Introduction

- Gaussian processes (GPs) are a non-parametric modeling technique used for regression and classification.
- They have many similarities with SVM, but they are based on probabilistic principles.
- GPs are random variables that have a joint Gaussian distribution.
- A GP specified by its mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$, where $\mathbf{x}$ represents input data points (i.e., the predictors).
- Formally, a GP can be written as:

$$f(\mathbf{x}) \sim \mathcal{GP}\{m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\}$$

GP Process Definitions

- $m(\mathbf{x})$ represents the expected value of the function $f(\mathbf{x})$ at the point $\mathbf{x}$. In many cases, it's assumed to be zero.
- The covariance or kernel function $k(\mathbf{x}, \mathbf{x}')$ measures the degree to which $f(\mathbf{x})$ and $f(\mathbf{x}')$ covary.
- The kernel function embodies the assumption about the function's smoothness.
- One commonly used kernel is the squared exponential (SE) or Gaussian kernel:

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2l^2}\right)$$

where σ^2 is the maximum covariance, and l is the length scale parameter that dictates the smoothness of the realized function.

Adding noise

- To this point, GP has been a representation of a function.
- In real scenarios, they're used to model and outcome $\boldsymbol{y}$ via

$$\boldsymbol{y} = f(\mathbf{x}) + \epsilon$$

where $\epsilon \sim N(0, \sigma_n^2)$ is independent of $f(\mathbf{x})$.

- By adding a noise term to the covariance function of $\boldsymbol{y}$ is:

$$k_y(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2l^2}\right) + \sigma_n^2 \delta_{\mathbf{x}\mathbf{x}'}$$

where δ is the Kronecker delta, which is 1 if $\mathbf{x} = \mathbf{x}'$ and 0 otherwise.

Prior and Posterior

- Prior to observing any data, we have a prior belief about f represented by a GP.
- Upon observing data $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$, the prior is combined with the data's likelihood to form the posterior distribution over f.
- If the prior is $\mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$ and the likelihood is Gaussian, the posterior over f given $\mathcal{D}$ is also a GP,
- The posterior mean and covariance functions calculated for a specific point, say $\mathbf{x}_*$.

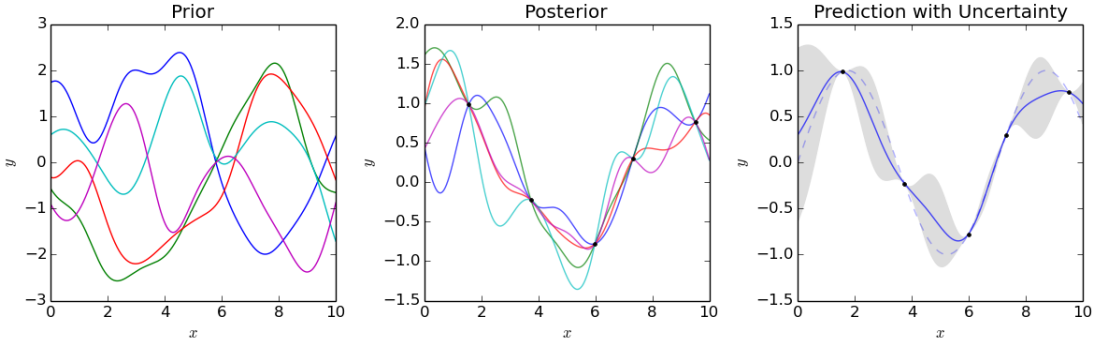
Predictions

- The posterior at $\mathbf{x}_*$, is given by:

$$\begin{aligned} m_{post}(\mathbf{x}_*) = \bar{f}_* &= \mathbf{k}_*^\top \{\mathbf{K}(\mathbf{x},\mathbf{x}) + \sigma_n^2 \mathbf{I}\}^{-1} \mathbf{y}, \\ k_{post}(\mathbf{x}_*, \mathbf{x}_*) = \mathbb{V}[f_*] &= k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{k}_*, \end{aligned}$$

  where $\mathbf{k}_* = [k(\mathbf{x}_*, \mathbf{x}_1), ..., k(\mathbf{x}_*, \mathbf{x}_n)]^\top$, $\mathbf{K}(\mathbf{x},\mathbf{x})$ is the covariance matrix of the training data, and
  - $\bar{f}_*$ is the posterior mean,
  - $\mathbb{V}[f_*]$ is the posterior variance (we could also get the posterior covariance).
- This forms the basis for making predictions with Gaussian processes, providing not only the expected value but also the uncertainty around that expectation.

Prior and Posterior



Hyperparameter Training

- The hyperparameters, e.g., σ^2, l, and σ_n^2 in the SE kernel, are usually optimized by maximizing the log marginal likelihood (LML) of the observed data with respect to the hyperparameters.
- The LML is given by:

$$\log p(\mathbf{y}|\mathbf{X}) = -\frac{1}{2}\mathbf{y}^\top(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K} + \sigma_n^2\mathbf{I}| - \frac{n}{2}\log(2\pi),$$

  where $|\cdot|$ denotes the determinant of a matrix.
- This optimization can be done using gradient descent or other optimization techniques.

Other Kernels

- **Exponential Kernel:** similar to the squared exponential kernel but less smooth

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||}{l}\right)$$

- **Periodic Kernel:** used for functions that exhibit periodicity.

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{2\sin^2(\pi||\mathbf{x} - \mathbf{x}'||/p)}{l^2}\right)$$

- **Linear Kernel:** where the function f is linear in the input space.

$$k(\mathbf{x}, \mathbf{x}') = \sigma_b^2 + \sigma_v^2(\mathbf{x} - \mathbf{c})^\top(\mathbf{x}' - \mathbf{c})$$

among many others.

GPs for classification

- Gaussian Processes (GPs) are primarily used for regression problems, but they can also be adapted for classification tasks.
- GP classification is a type of probabilistic classification.
- To use GPs for classification, we'll state the model in a latent framework.
- We assume there exists a latent function $f(x)$ that gives rise to the observed class labels.
- However, $f(x)$ nor $f(x) + \epsilon$ are not observed.

GPs for classification

- For binary classification, the class label y is generated from $f(x)$ through a probabilistic mechanism.
- For instance, $y = 1$ with probability $p = \sigma\{f(x)\}$ and $y = 0$ with probability $1 - p$, where σ is the logistic sigmoid function squashing $f(x)$ into the interval $(0, 1)$.
- This defines the likelihood function:

$$p(y|f(x)) = \sigma\{f(x)\}^y \left[1 - \sigma\{f(x)\}\right]^{(1-y)}$$

- The goal is to infer the posterior distribution of the latent function given the observed data, $p(f(x)|\mathbf{y}, \mathbf{x})$, where $\mathbf{y}$ are the labels and $\mathbf{x}$ are the input features.
- Approximate inference techniques – Laplace, Expectation Propagation, Variational Bayes – are used.

Connection to SVM

- The core mathematical connection between SVMs and GPs lies in their use of kernel functions.
- Both methods can be formulated in a high-dimensional feature space implied by a kernel function, and they both depend critically on the choice of this kernel.
  - SVMs $\rightarrow$ the kernel trick enables them to construct a hyperplane in a high-dimensional feature space corresponding to a non-linear decision boundary in the original input space.
  - GPs $\rightarrow$ the kernel (or covariance) function defines the covariance structure of the data, or informally, the similarity measure between points in the input space.

Connection to SVM

- The core mathematical connection between SVMs and GPs lies in their use of kernel functions.
- Both methods can be formulated in a high-dimensional feature space implied by a kernel function, and they both depend critically on the choice of this kernel.
  - SVMs $\rightarrow$ the kernel trick enables them to construct a hyperplane in a high-dimensional feature space corresponding to a non-linear decision boundary in the original input space.
  - GPs $\rightarrow$ the kernel (or covariance) function defines the covariance structure of the data, or informally, the similarity measure between points in the input space.
- Probabilistic Interpretation:
  - SVMs: Typically, SVMs don't provide a probability distribution over the outputs. However, they can be adapted to provide probabilistic outputs (e.g., by fitting a logistic regression model to the outputs of the SVM).
  - GPs: In contrast, GPs provide a full probabilistic view.

The good of GPs

- Non-parametric.
- Predictive Uncertainty.
- Kernel Flexibility.
- Interpretability.
- Fewer Hyperparameters.

The not so good of GPs

- Computational Complexity
- Choice of Kernel
- Local Minima in Hyperparameter Optimization
- Sensitivity to Noise Level
- Scalability: Standard GPs don't scale well with dimensionality. As the input space's dimension increases, the volume of the space increases exponentially, and more data are needed to provide the same level of model fidelity. This phenomenon is known as the "curse of dimensionality."