Support Vector Machines for Classificaiton

Justin Silverman

02/23/2021

Table of Contents

Sources

A collection of slides in this talk are taken from the following sources: - Chapter 14 of Murphy

Section 1

The Kernel Trick

Kernels

A Kernel function is a real-valued function of two arguments, $K(x_i, x_j) \in \mathcal{R}$. Typically the function is symmetric (i.e., $K(x_i, x_j) = K(x_j, x_i)$) and non-negative (i.e., $K(x_i, x_j) \geq 0$) such that K can be interpreted as a measure of similarity between x_i and x_j but this is not required.

If K is symmetric and non-negative than any **Gram matrix*** Σ of K on a collection of points $(x_1, \dots, x_N)$ defined as

$$\Sigma_{ij} == K(x_i, x_j)$$

is symmetric positive definite (e.g., a covariance matrix).

Types of Kernels

- Linear Kernels
- RBF Kernels (aka Gaussian Kernels or Squared Exponential)
- Mercer Kernels
- Rational Quadratic Kernels
- Matern Kernels
- String Kernels
- Pyramid Match Kernels (used in computer vision)
- Fisher Kernels
- and many more. . .

Kernel Machines

A GLM where we choose a set of *centroids* $(\mu_1, \ldots, \mu_k)$ and define basis functions (aka feature vectors, aka covariates) to have the form

$$\phi(x) = [K(x, \mu_1), \ldots, K(x, \mu_k)].$$

Kernel Machines

A GLM where we choose a set of *centroids* $(\mu_1, \ldots, \mu_k)$ and define basis functions (aka feature vectors, aka covariates) to have the form

$$\phi(x) = [K(x, \mu_1), \ldots, K(x, \mu_k)].$$

If K is an RBF kernel this is called an **RBF Network**.

This is nice and permits non-linear extensions of all kinds of linear or genearlized linear models.

Kernel Machines

A GLM where we choose a set of *centroids* $(\mu_1, \ldots, \mu_k)$ and define basis functions (aka feature vectors, aka covariates) to have the form

$$\phi(x) = [K(x, \mu_1), \ldots, K(x, \mu_k)].$$

If K is an RBF kernel this is called an **RBF Network**.

This is nice and permits non-linear extensions of all kinds of linear or genearlized linear models.

But how are the *centroids* $(\mu_1, \ldots, \mu_k)$ **picked?** In high-dimensional settings, you essentially need to tile the space and k will be too large to be useful.

An alternative is to use the kernel trick.

An Alternative

A simpler approach is to simply pick the centroids to be the training data so that the basis functions you feed into the model have the form

$$\phi(x) = [K(x, x_1), \ldots, K(x, x_N)].$$

An Alternative

A simpler approach is to simply pick the centroids to be the training data so that the basis functions you feed into the model have the form

$$\phi(x) = [K(x, x_1), \ldots, K(x, x_N)].$$

If N is very large this too can be computationally challenging. **Sparse Kernel Machines** take this approach but efficiently select only a subset of the training examples.

Support Vector Machines are an example of a Sparse Kernel Machine.

The Kernel Trick

Rather than plugging $\phi(x) = [K(x, x_1), \ldots, K(x, x_N)]$ into a standard linear / generalized linear modeling framework, we can instead change the algorithm.

Instead of ever having the model use x's directly, we instead replace all inner products in the original algorithm (things of the form $\underline{< x, x_i >}$) with a call to the kernel function $K(x, x_i)$.

$x^T x$

This is called the **Kernel Trick** and we will see it a few times throughout this course. *(We already saw it, although it was somewhat implicit, with Gaussian Processes.)*

$$Y \sim N(\underset{x^T x}{\mu}, \boxed{\boxed{\sigma^2 X^T X} + \sigma^2 I}$$

The Kernel Trick

Rather than plugging $\phi(x) = [K(x, x_1), \ldots, K(x, x_N)]$ into a standard linear / generalized linear modeling framework, we can instead change the algorithm.

Instead of ever having the model use x's directly, we instead replace all inner products in the original algorithm (things of the form $< x, x_i >$) with a call to the kernel function $K(x, x_i)$.

This is called the **Kernel Trick** and we will see it a few times throughout this course. *(We already saw it, although it was somewhat implicit, with Gaussian Processes.)*

Keep this idea in the back of your mind as we transition to SVMs.

Section 2

Support Vector Machines in Context

Support Vector Machines in Context

We have already seen models in their Loss representation of the form:

$$\underline{J(w,\lambda)} = \sum_{i=1}^{N} \underline{L(y_i, \hat{y}_i)} + \lambda||w||^2.$$

- If L is quadratic loss we get Ridge Regression
- If L is log-loss we get Penalized Logistic Regression

Support Vector Machines in Context

We have already seen models in their Loss representation of the form:

$$J(w, \lambda) = \sum_{i=1}^{N} L(y_i, \hat{y}_i) + \lambda ||w||^2.$$

- If L is quadratic loss we get Ridge Regression
- If L is log-loss we get Penalized Logistic Regression

As an example: for ridge regression this leads to estimators for w of the form $\boxed{\hat{w} = (\underline{X^T X} + \lambda I)^{-1} X^T y}$ and predictions of the form $\hat{w}_0 + \hat{w}x$, both of which can be written in such a way that they only involve inner products of the form $\underline{x^T x^*}$ (for some evaluation point x^*).

Support Vector Machines in Context

We have already seen models in their Loss representation of the form:

$$J(w, \lambda) = \sum_{i=1}^{N} L(y_i, \hat{y}_i) + \lambda ||w||^2.$$

- If L is quadratic loss we get Ridge Regression
- If L is log-loss we get Penalized Logistic Regression

As an example: for ridge regression this leads to estimators for w of the form $\hat{w} = (X^T X + \lambda I)^{-1} X^T y$ and predictions of the form $\hat{w}_0 + \hat{w} x$, both of which can be written in such a way that they only involve inner products of the form $x^T x^*$ (for some evaluation point x^*).

This can then be kernelized by replacing those inner products with calls to $K(x, x^*)$.

This results in a kernel machine. SVMs are this plus a modification of the loss function (e.g., the Hinge Loss) such that predictions only depend on a subset of the training points e.g., a **Sparse Kernel Machine**.

•

Section 3

A Transition to A Different View of SVMs

A Transition to A Different View of SVMs

You should understand, in the abstract, where SVMs fit into the greater category of machine learning methods. Understand the general concepts in the prior slides.

However, going much further gets into some weeds of theory and for now I want to transition to a different way of viewing these models that will be useful when you are doing applied work.

Those wishing to learn more are encouraged to read Chapter 12 of Hastie, Tibshirani, and Friedman.