Sparse Kernel Machines

Chapter 7

Prof. Reza Azadeh

University of Massachusetts Lowell

Previously

- In previous chapter, we looked at non-parametric methods in which we have to evaluate a kernel for all pairs of data points.
- This process can be time consuming and computationally expensive.
- In this chapter, we look at kernel methods with sparse solutions, in which for the prediction of new points we only need to evaluate the kernel with a subset of points.

Background - Lagrange Multipliers (1)

Consider the problem of finding the maximum of a function $f(x_1, x_2)$ subject to a constraint relating x_1 and x_2.

$$\begin{aligned} \text{maximize } & f(\mathbf{x}) \\ \text{s.t. } & g(\mathbf{x}) = 0 \end{aligned}$$

Background - Lagrange Multipliers (2)

To approach this problem, we introduce a parameter λ called a Lagrange multiplier and write the Lagrangian function as

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x})$$

We then need to find the stationary point of $L(\mathbf{x}, \lambda)$ with respect to both $\mathbf{x}$ and λ.

Background - Lagrange Multipliers example (1)

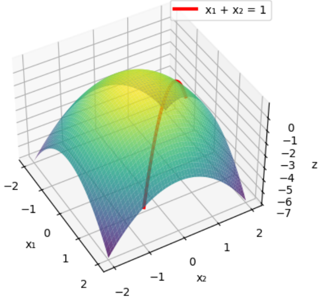
Suppose we would like to solve

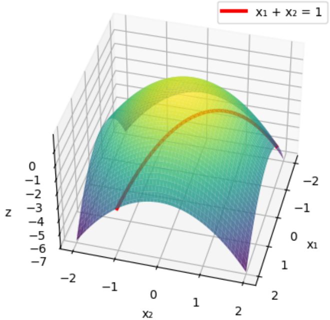
$$\begin{aligned} \text{maximize } & f(x_1, x_2) = 1 - x_1^2 - x_2^2 \\ \text{s.t. } & g(x_1, x_2) = x_1 + x_2 - 1 = 0 \end{aligned}$$

We write the Lagrangian function

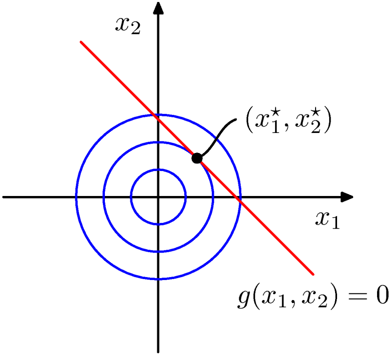
$$L(\mathbf{x}, \lambda) = 1 - x_1^2 - x_2^2 + \lambda(x_1 + x_2 - 1)$$

Background - Lagrange Multipliers example (2)





Background - Lagrange Multipliers example (2)



Background - Lagrange Multipliers example (3)

We then get the gradients

$$\frac{\partial L}{\partial x_1} = -2x_1 + \lambda = 0$$
$$\frac{\partial L}{\partial x_2} = -2x_2 + \lambda = 0$$
$$\frac{\partial L}{\partial \lambda} = x_1 + x_2 + 1 = 0$$

Solving this set of equations results in the stationary point $(x_1^*, x_2^*) = (\frac{1}{2}, \frac{1}{2})$.

Maximum Margin Classifiers (1)

Consider a 2-class classification problem using the following linear model:

$$y(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}) + b$$

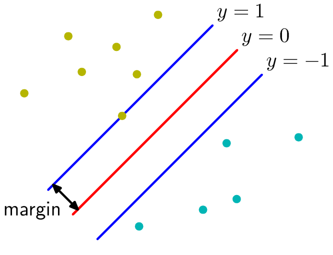
where $\boldsymbol{\phi}(\mathbf{x})$ denotes a fixed feature space transformation, and we have made the bias parameter explicit.
We also consider a training data set with N input vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N$ with corresponding target values $t_1, \ldots, t_N$ where $t_n = \{-1, 1\}$, and new data point $\mathbf{x}$ are classified according to the sign of $y(\mathbf{x})$.

Maximum Margin Classifiers (2)

We also assume that the training set is linearly separable in feature space, so there is at least one choice of parameters $\mathbf{w}$ and b, such that we get $y(\mathbf{x}_n) > 0$ for points with $t_n = +1$ and $y(\mathbf{x}_n) < 0$ for points with $t_n = -1$. And therefore for all data points we have $t_n y(\mathbf{x}_n) > 0$.

The **support vector machine** approaches this problem using the concept of the *margin*, which is defined to be the smallest distance between the decision boundary and any of the samples.

Maximum Margin Classifiers (3)



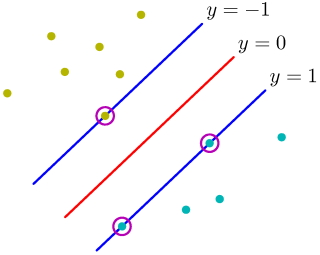
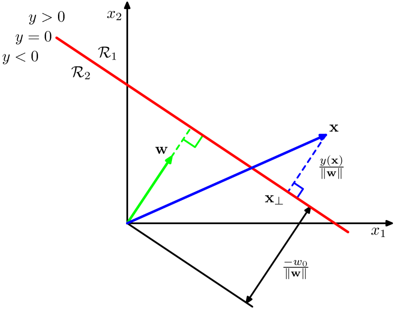


Figure: The margin is defined as the perpendicular distance between the decision boundary and the closest of the data points (left). Maximizing the margin leads to a particular choice of decision boundary (right). The location of the boundary is determined by a **subset** of the data points, known as *support vectors*, shown by circles.

Recall from our discussions on linear models for classification, the perpendicular distance of a point $\mathbf{x}$ from a hyper-plane defined by $y(\mathbf{x}) = 0$ is given by $|y(\mathbf{x})|/\|\mathbf{w}\|$.



Maximum Margin Classifiers (5)

In this section, we are interested in the solution where all the data points are correctly classified. Thus the distance of a point $\mathbf{x}_n$ to the decision surface is given by

$$\frac{t_n y(\mathbf{x})}{\|\mathbf{w}\|} = \frac{t_n(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) + b)}{\|\mathbf{w}\|}$$

We wish to optimize the parameters $\mathbf{w}$ and b to maximize the distance. Thus the maximum margin solution is found by solving

$$\mathrm{argmax}_{\mathbf{w},b}\left\{\frac{1}{\|\mathbf{w}\|}\min_n\left[t_n(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) + b)\right]\right\}$$

Maximum Margin Classifiers (6)

Since solving this optimization problem is difficult, we convert it to a simpler formulation. We do this simplification based on the knowledge that scaling $\mathbf{w} \to \kappa\mathbf{w}$ and $b \to \kappa b$ does not change $t_n y(\mathbf{x}_n)/\|\mathbf{w}\|$. We can use this freedom to set

$$t_n(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) + b) = 1$$

for the point that is closest to the surface. All the data points then satisfy

$$t_n(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) + b) \geq 1, \quad n = 1, \ldots, N$$

Maximum Margin Classifiers (7)

Using this approach, the optimization problem requires that we maximize $\|\mathbf{w}\|^{-1}$ which is equivalent to minimizing $\|\mathbf{w}\|^2$, and so we have to solve

$$\begin{aligned}&\operatorname{argmin}_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \\ &\text{s.t. } t_n(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) + b) \geq 1\end{aligned}$$

This problem can be solved using the Lagrange multiplier method, with multipliers $a_n \geq 0$ with one multiplier a_n for each of the constraints, giving

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n\{t_n(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n) + b) - 1\}$$

Maximum Margin Classifiers (8)

Setting the derivatives w.r.t $\mathbf{w}$ and b equal to zero, we obtain

$$\mathbf{w} = \sum_{n=1}^{N} a_n t_n \boldsymbol{\phi}(\mathbf{x}_n)$$

$$0 = \sum_{n=1}^{N} a_n t_n.$$

Maximum Margin Classifiers (9)

Eliminating $\mathbf{w}$ and b from the Lagrangian, we should maximize

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

with respect to $\mathbf{a}$ subject to

$$a_n \geq 0, \quad n = 1, \ldots, N$$

$$\sum_{n=1}^{N} a_n t_n = 0.$$

Note that the kernel is defined by $k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')$

Maximum Margin Classifiers (10)

To classify a new point using the trained model, we evaluate the sign of $y(\mathbf{x})$ as

$$y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b$$

The points that are on the maximum margin hyperplane in feature space, a.k.a *support vectors*, surface will appear in this formula but all other points do not because they satisfy $a_n = 0$. This means after training, a significant portion of the data points can be discarded and only the support vectors retained.

Maximum Margin Classifiers (11)

After finding $\mathbf{a}$, we can also find b by solving

$$b = \frac{1}{N_{\mathcal{S}}} \sum_{n \in \mathcal{S}} (t_n - \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m))$$

Maximum Margin Classifiers (12)

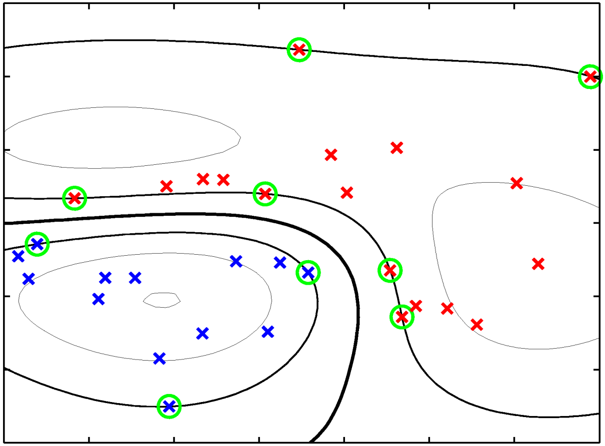


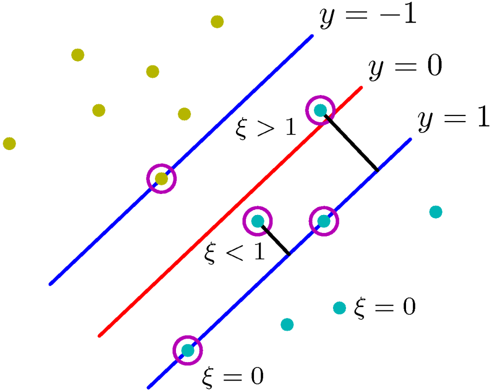
Figure: Example of a classification resulting from training a support vector machine on a simple synthetic data set using a Gaussian kernel. Also shown are the decision boundaries, the margin boundaries, and the support vectors.

Overlapping class distributions (1)

- When the class-conditional distributions overlap, exact separation of the training data can lead to poor generalization.
- We can do this by allowing data points to be on the "wrong side" of the margin boundary, but with a penalty that increases with the distance from that boundary.
- We introduce the *slack variable* $\xi_n \geq 0$ where $n = 1, \dots, N$ with one slack variable for each data point.
- These are defined by $\xi_n = 0$ for points that are on or inside the correct margin boundary and $\xi_n = |t_n - y(\mathbf{x}_n)|$ for other points.
- a data point on the decision boundary will have $\xi_n = 1$.

Overlapping class distributions (2)

- Data points with $\xi_n > 1$ will be misclassified.



Overlapping class distributions (3)

Our goal now is to maximize the margin while softly penalizing points that lie on the wrong side of the margin boundary.

$$\text{minimize } C \sum_{n=1}^{N} \xi_n + \frac{1}{2} \|\mathbf{w}\|^2$$

$$\text{s.t. } t_n y(\mathbf{x}_n) \geq 1 - \xi_n$$

where the parameter C controls the trade-off between the slack variable penalty and the margin.

Overlapping class distributions (4)

This can be done similar to the previous solution by forming the Lagrangian multiplier, resulting in

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

which is identical to the separable case, except that the constraints are different. We should maximize the above equation w.r.t $\mathbf{a}$ subject to the following *box constraints*

$$0 \leq a_n \leq C$$

$$\sum_{n=1}^{N} a_n t_n = 0$$

Overlapping class distributions (5)

We can also find b using

$$b = \frac{1}{N_{\mathcal{M}}} \sum_{n \in \mathcal{M}} (t_n - \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m))$$

where $\mathcal{M}$ denotes the set of indices of data points having $0 < a_n < C$.

ν-SVM

An alternative, equivalent formulation of the support vector machine, known as ν-SVM involves solving

$$\text{maximize } \tilde{L}(\mathbf{a}) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

$$\text{s.t. } 0 \leq a_n \leq \frac{1}{N}$$

$$\sum_{n=1}^{N} a_n t_n = 0$$

$$\sum_{n=1}^{N} a_n \geq \nu$$

ν-SVM - example

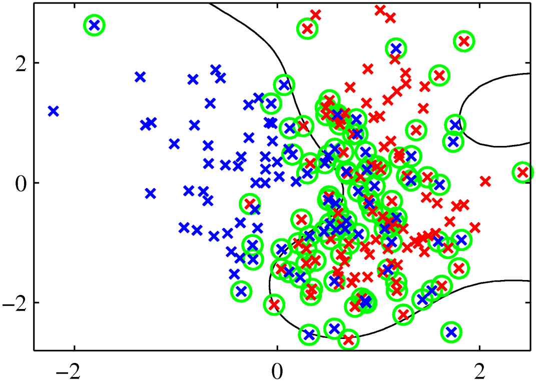


Figure: ν-SVM applied to a non-separable data set in two dimensions. The support vectors are indicated by circles.

Multiclass SVMs

- The support vector machine is fundamentally a two-class classifier.
- There exist methods that use a single objective for training all K SVMs simultaneously, based on maximizing the margin from each to remaining classes. $\rightarrow$ can be slow, but the most commonly used method.
- Another approach is to training $K(K-1)/2$ different two-class SVMs on all the possible pairs of classes, and then to classify test points according to which class has the highest number of votes. $\rightarrow$ still slow

SVMs for regression

To extend SVMs for regression, we replace the regularized error function

$$\frac{1}{2}\sum_{n=1}^{N}\{y_n - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

by an ϵ-insensitive error function, which gives zero error if the absolute difference between the predictions $y(\mathbf{x})$ and the target t is less than ϵ where $\epsilon > 0$.

$$E_\epsilon(y(\mathbf{x}) - t) = \begin{cases} 0, & \text{if } |y(\mathbf{x}) - t| < \epsilon \\ |y(\mathbf{x}) - t| - \epsilon, & \text{otherwise} \end{cases}$$

ϵ-insensitive error function

$$E_\epsilon(y(\mathbf{x}) - t) = \begin{cases} 0, & \text{if } |y(\mathbf{x}) - t| < \epsilon \\ |y(\mathbf{x}) - t| - \epsilon, & \text{otherwise} \end{cases}$$

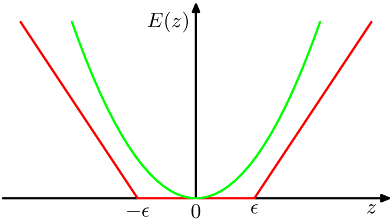


Figure: quadratic error function (green) and ϵ-insensitive error function (red)

Minimization problem

So in SVM for regression, instead of

$$\frac{1}{2}\sum_{n=1}^{N}\{y_n - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

we minimize

$$C\sum_{n=1}^{N} E_\epsilon(y(\mathbf{x}_n) - t_n) + \frac{1}{2}\|\mathbf{w}\|^2$$

where $y(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}) + b$.

ϵ-insensitive tube

We introduce two slack variables for each data point $\xi_n \geq 0$ and $\hat{\xi}_n \geq 0$. This definition results in a ϵ-insensitive *tube*.

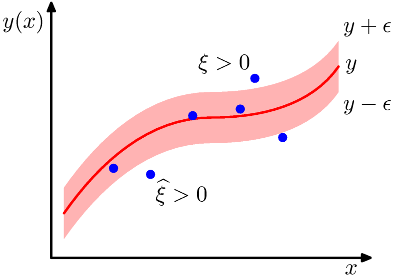


Figure: regression curve together with the ϵ-tube. Points above the tube have $\xi > 0$ and $\hat{\xi} = 0$, whereas points below the tube have $\xi = 0$ and $\hat{\xi} > 0$, and points inside the tube have $\xi = \hat{\xi} = 0$.

The error function then can be re-written as

$$C\sum_{n=1}^{N}(\xi_n+\hat{\xi}_n)+\frac{1}{2}\|\mathbf{w}\|^2$$

To cover all the constraints, we form the Lagrange multiplier as

$$\begin{aligned}L = C\sum_{n=1}^{N}(\xi_n+\hat{\xi}_n)+\frac{1}{2}\|\mathbf{w}\|^2-\sum_{n=1}^{N}(\mu_n\xi_n+\hat{\mu}_n\hat{\xi}_n)\\ -\sum_{n=1}^{N}a_n(\epsilon+\xi_n+y_n-t_n)-\sum_{n=1}^{N}\hat{a}_n(\epsilon+\hat{\xi}_n-y_n+t_n)\end{aligned}$$

Lagrange multiplier

We then calculate the derivatives of the Lagrangian w.r.t $\mathbf{w}, b, \xi_n$, and $\hat{\xi}_n$ and setting them to zero.

$$\tilde{L}(\mathbf{a}, \hat{\mathbf{a}}) = -\frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} (a_n - \hat{a}_n)(a_m - \hat{a}_m) k(\mathbf{x}_n, \mathbf{x}_m) \\ - \epsilon \sum_{n=1}^{N} (a_n + \hat{a}_n) + \sum_{n=1}^{N} (a_n - \hat{a}_n) t_n$$

where $k(\mathbf{x}_n, \mathbf{x}_m) = \boldsymbol{\phi}(\mathbf{x}_n)^\top \boldsymbol{\phi}(\mathbf{x}_m)$, and the prediction

$$y(\mathbf{x}) = \sum_{n=1}^{N} (a_n - \hat{a}_n) k(\mathbf{x}, \mathbf{x}_n) + b$$

ν-SVM for Regression

Similar to the classification case, there is an alternative formulation that involves maximizing

$$\tilde{L}(\mathbf{a}, \hat{\mathbf{a}}) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}(a_n - \hat{a}_n)(a_m - \hat{a}_m)k(\mathbf{x}_n, \mathbf{x}_m) + \sum_{n=1}^{N}(a_n - \hat{a}_n)t_n$$

$$\begin{aligned} \text{s.t. } \; & 0 \leq a_n \leq C/N \\ & 0 \leq \hat{a}_n \leq C/N \\ & \sum_{n=1}^{N}(a_n - \hat{a}_n) = 0 \\ & \sum_{n=1}^{N}(a_n + \hat{a}_n) \leq \nu C \end{aligned}$$

ν-SVM for Regression - example

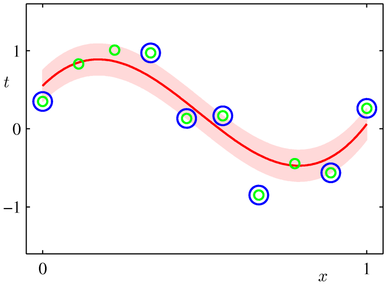


Figure: ν-SVM for regression applied to the sinusoidal synthetic data set using Gaussian kernel. The predicted regression curve (red), the ϵ-tube (shaded region), support vectors (blue circles) other data points (green circles).