18-661 Introduction to Machine Learning

SVM – III

Fall 2020

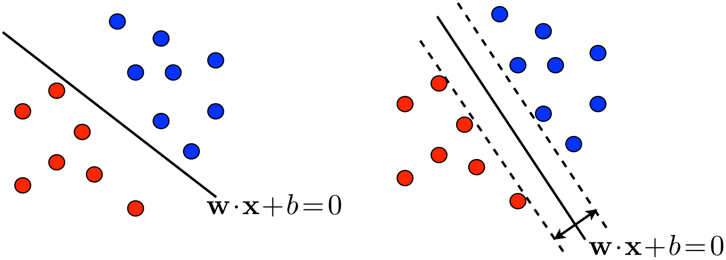
ECE – Carnegie Mellon University

Outline

1. Review of SVM Max Margin Formulation
2. A Dual View of SVMs (the short version)
3. Dual Derivation of SVMs (optional)
4. Kernel SVM

Review of SVM Max Margin Formulation

Intuition: Where to put the decision boundary?



Idea: Find a decision boundary in the '*middle*' of the two classes that:

- Perfectly classifies the training data
- Is as far away from every training point as possible

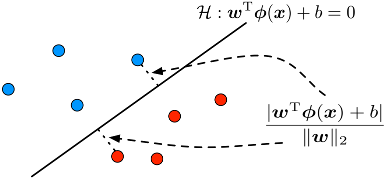
Let us apply this intuition to build a classifier that MAXIMIZES THE MARGIN between training points and the decision boundary

Defining the Margin

Margin

Smallest distance between the hyperplane and all training points

$$\text{MARGIN}(\boldsymbol{w}, b) = \min_n \frac{y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b]}{\|\boldsymbol{w}\|_2}$$



Rescaled Margin to Avoid Scaling Ambiguity

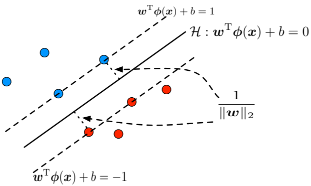
We can further constrain the problem by scaling $(\boldsymbol{w}, b)$ such that

$$\min_n y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] = 1$$

We've fixed the numerator in the $\textsc{margin}(\boldsymbol{w}, b)$ equation, and we have:

$$\textsc{margin}(\boldsymbol{w}, b) = \frac{\min_n y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b]}{\|\boldsymbol{w}\|_2} = \frac{1}{\|\boldsymbol{w}\|_2}$$

Hence the points closest to the decision boundary are at distance $\frac{1}{\|\boldsymbol{w}\|_2}$!



SVM: max margin formulation for separable data

Assuming separable training data, we thus want to solve:

$$\max_{\boldsymbol{w},b} \underbrace{\frac{1}{\|\boldsymbol{w}\|_2}}_{\text{margin}} \quad \text{such that} \quad \underbrace{y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1, \quad \forall \quad n}_{\text{scaling of } \boldsymbol{w},\, b}$$

This is equivalent to

$$\begin{aligned} \min_{\boldsymbol{w},b} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 \\ \text{s.t.} \quad & y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1, \quad \forall \quad n \end{aligned}$$

Given our geometric intuition, SVM is called a **max margin** (or large margin) classifier. The constraints are called **large margin constraints**.

SVM for non-separable data

Constraints in separable setting

$$y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1, \quad \forall \quad n$$

Constraints in non-separable setting

Idea: modify our constraints to account for non-separability! Specifically, we introduce *slack variables* $\xi_n \geq 0$:

$$y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n, \quad \forall \quad n$$

- For "hard" training points, we can increase ξ_n until the above inequalities are met
- What does it mean when ξ_n is very large?

Soft-margin SVM formulation

We do not want ξ_n to grow too large, and we can control their size by incorporating them into our optimization problem:

$$\begin{aligned} \min_{\boldsymbol{w}, b, \boldsymbol{\xi}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C \sum_n \xi_n \\ \text{s.t.} \quad & y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n, \quad \forall \ \ n \\ & \xi_n \geq 0, \quad \forall \ n \end{aligned}$$

What is the role of C?

- User-defined hyperparameter
- Trades off between the two terms in our objective
- Same idea as the regularization term in ridge regression

How to solve this problem?

$$\begin{aligned}\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n \\ \text{s.t.} \quad & y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n, \quad \forall \;\; n \\ & \xi_n \geq 0, \quad \forall \; n\end{aligned}$$

- This is a *convex quadratic program*: the objective function is quadratic in $\boldsymbol{w}$ and linear in $\boldsymbol{\xi}$ and the constraints are linear (inequality) constraints in $\boldsymbol{w}$, b and ξ_n.
- We can solve the optimization problem using general-purpose solvers, e.g., Matlab's `quadprog()` function.

A Dual View of SVMs (the short version)

What is duality?

Duality is a way of transforming a constrained optimization problem.

It tells us sometimes-useful information about the problem structure, and can sometimes make the problem easier to solve.

- Dual problem is always convex–easy to solve.
- Primal and dual problems are not always equivalent.
- Dual variables tell us “how bad” constraints are.

The main point you should understand is that we will solve the dual SVM problem in lieu of the max margin (primal) formulation

Derivation of the dual

Here is a skeleton of how to derive the dual problem.

Recipe

1. Formulate the generalized Lagrangian function that incorporates the constraints and introduces dual variables
2. Minimize the Lagrangian function over the primal variables
3. Substitute the primal variables for dual variables in the Lagrangian
4. Maximize the Lagrangian with respect to dual variables
5. Recover the solution (for the primal variables) from the dual variables

Deriving the dual for SVM

Primal SVM

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n$$
$$\text{s.t.} \quad y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n, \quad \forall \quad n$$
$$\xi_n \geq 0, \quad \forall\ n$$

The constraints are equivalent to the following canonical forms:

$$-\xi_n \leq 0 \quad \text{and} \quad 1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] - \xi_n \leq 0$$

Lagrangian

$$L(\boldsymbol{w}, b, \{\xi_n\}, \{\alpha_n\}, \{\lambda_n\}) = C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \lambda_n \xi_n + \sum_n \alpha_n\{1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] - \xi_n\}$$

under the constraints that $\alpha_n \geq 0$ and $\lambda_n \geq 0$.

Deriving the dual of SVM

Lagrangian

$$L(\boldsymbol{w}, b, \{\xi_n\}, \{\alpha_n\}, \{\lambda_n\}) = C \sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \lambda_n \xi_n + \sum_n \alpha_n \{1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] - \xi_n\}$$

under the constraints that $\alpha_n \geq 0$ and $\lambda_n \geq 0$.

- Primal variables: $\boldsymbol{w}$, $\{\xi_n\}$, b; dual variables $\{\lambda_n\}$, $\{\alpha_n\}$
- Minimize the Lagrangian function over the primal variables by setting $\frac{\partial L}{\partial \boldsymbol{w}} = 0$, $\frac{\partial L}{\partial b} = 0$, and $\frac{\partial L}{\partial \xi_n} = 0$.
- Substitute the solutions to primal variables for dual variables in the Lagrangian
- Maximize the Lagrangian with respect to dual variables
- After some further maths and simplifications, we have...

Dual formulation of SVM

Dual is also a convex quadratic program

$$\max_{\boldsymbol{\alpha}} \quad \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{x}_m^\top \boldsymbol{x}_n$$
$$\text{s.t.} \quad 0 \le \alpha_n \le C, \quad \forall\ n$$
$$\sum_n \alpha_n y_n = 0$$

- There are N dual variables α_n, one for each data point
- Independent of the size d of $\mathbf{x}$: SVM scales better for high-dimensional feature.
- May seem like a lot of optimization variables when N is large, but many of the α_n's become zero. α_n is non-zero only if the n^{th} point is a support vector

Once we solve for α_n's, how to get w and b?

Recovering $\mathbf{w}$

$$\frac{\partial L}{\partial \boldsymbol{w}} = 0 \rightarrow \boldsymbol{w} = \sum_n \alpha_n y_n \boldsymbol{x}_n$$

Only depends on support vectors, i.e., points with $\alpha_n > 0$!

Recovering b

If $0 < \alpha_n < C$ and $y_n \in \{-1, 1\}$:

$$\begin{aligned} y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] = 1 \\ b &= y_n - \boldsymbol{w}^\top \boldsymbol{x}_n \\ b &= y_n - \sum_m \alpha_m y_m \boldsymbol{x}_m^\top \boldsymbol{x}_n \end{aligned}$$

Why do many α_n's become zero?

$$\max_{\boldsymbol{\alpha}} \quad \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{x}_m^\top \boldsymbol{x}_n$$

$$\text{s.t.} \quad 0 \leq \alpha_n \leq C, \quad \forall\ n$$

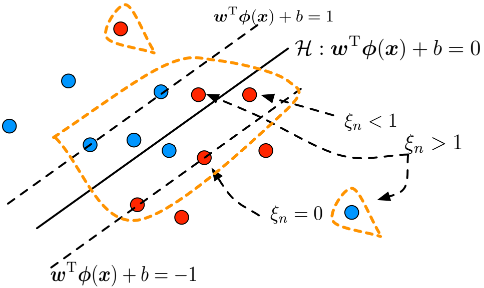
$$\sum_n \alpha_n y_n = 0$$

- By strong duality and KKT complementary slackness conditions, it tells us:

$$\alpha_n\{1 - \xi_n - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b]\} = 0 \quad \forall n$$

- This tells us that $\alpha_n > 0$ iff $1 - \xi_n = y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b]$
  - If $\xi_n = 0$, then support vector is on the margin
  - Otherwise, $\xi_n > 0$ means that the point is an outlier

Visualizing the support vectors



Support vectors ($\alpha_n > 0$) are highlighted by the dotted orange lines.

- $\xi_n = 0$ and $0 < \alpha_n < C$ when $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] = 1$.
- $\xi_n > 0$ and $\alpha_n = 0$ if $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] < 1$.

Outline

1. Review of SVM Max Margin Formulation
2. A Dual View of SVMs (the short version)
3. Dual Derivation of SVMs (optional)
4. Kernel SVM

Dual Derivation of SVMs (optional)

Derivation of the dual

We will next derive the dual formulation for SVMs.

Recipe

1. Formulate the generalized Lagrangian function that incorporates the constraints and introduces dual variables
2. Minimize the Lagrangian function over the primal variables
3. Substitute the primal variables for dual variables in the Lagrangian
4. Maximize the Lagrangian with respect to dual variables
5. Recover the solution (for the primal variables) from the dual variables

Deriving the dual for SVM

Primal SVM

$$\begin{aligned}\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n \\ \text{s.t.} \quad & y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n, \quad \forall \quad n \\ & \xi_n \geq 0, \quad \forall \ n\end{aligned}$$

The constraints are equivalent to $-\xi_n \leq 0$ and $1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] - \xi_n \leq 0$.

Lagrangian

$$\begin{aligned}L(\boldsymbol{w}, b, \{\xi_n\}, \{\alpha_n\}, \{\lambda_n\}) = C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \lambda_n \xi_n \\ + \sum_n \alpha_n\{1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] - \xi_n\}\end{aligned}$$

under the constraints that $\alpha_n \geq 0$ and $\lambda_n \geq 0$.

Minimizing the Lagrangian

Taking derivatives with respect to the primal variables

$$\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial}{\partial \boldsymbol{w}} \left(\frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \alpha_n y_n \boldsymbol{w}^\top \boldsymbol{x}_n \right) = \boldsymbol{w} - \sum_n y_n \alpha_n \boldsymbol{x}_n = 0$$
$$\frac{\partial L}{\partial b} = \frac{\partial}{\partial b} - \sum_n \alpha_n y_n b = -\sum_n \alpha_n y_n = 0$$
$$\frac{\partial L}{\partial \xi_n} = \frac{\partial}{\partial \xi_n} (C - \lambda_n - \alpha_n)\xi_n = C - \lambda_n - \alpha_n = 0$$

These equations link the primal variables and the dual variables and provide new constraints on the dual variables:

$$\boldsymbol{w} = \sum_n y_n \alpha_n \boldsymbol{x}_n$$
$$\sum_n \alpha_n y_n = 0$$
$$C - \lambda_n - \alpha_n = 0$$

Rearrange the Lagrangian

$$L(\cdot) = C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \lambda_n\xi_n + \sum_n \alpha_n\{1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] - \xi_n\}$$

where $\alpha_n \geq 0$ and $\lambda_n \geq 0$. We now know that $\boldsymbol{w} = \sum_n y_n\alpha_n\boldsymbol{x}_n$.

$$\begin{aligned}
g(\{\alpha_n\},\{\lambda_n\}) &= L(\boldsymbol{w}, b, \{\xi_n\}, \{\alpha_n\}, \{\lambda_n\}) \\
&= \underbrace{\sum_n (C - \alpha_n - \lambda_n)\xi_n}_{\text{gather terms with } \xi_n} + \frac{1}{2}\|\underbrace{\sum_n y_n\alpha_n\boldsymbol{x}_n}_{\text{substitute for } \boldsymbol{w}}\|_2^2 + \sum_n \alpha_n \\
&\quad - \sum_n \alpha_n y_n \underbrace{\left(\sum_m y_m\alpha_m\boldsymbol{x}_m\right)^\top}_{\text{again substitute for } \boldsymbol{w}} \boldsymbol{x}_n - \left(\sum_n \alpha_n y_n\right) b
\end{aligned}$$

Then, set $\sum_n \alpha_n y_n = 0$ and $C - \lambda_n - \alpha_n = 0$ and simplify to get..

Incorporate the constraints

Constraints from partial derivatives: $\sum_n \alpha_n y_n = 0$ and $C - \lambda_n - \alpha_n = 0$.

$$
\begin{aligned}
g(\{\alpha_n\},\{\lambda_n\}) &= L(\boldsymbol{w}, b, \{\xi_n\}, \{\alpha_n\}, \{\lambda_n\}) \\
&= \sum_n \underbrace{(C - \alpha_n - \lambda_n)}_{\text{equal to 0!}} \xi_n + \frac{1}{2} \| \sum_n y_n \alpha_n \boldsymbol{x}_n \|_2^2 + \sum_n \alpha_n \\
&- \underbrace{\left(\sum_n \alpha_n y_n \right)}_{\text{equal to 0!}} b - \sum_n \alpha_n y_n \left(\sum_m y_m \alpha_m \boldsymbol{x}_m \right)^\top \boldsymbol{x}_n \\
&= \sum_n \alpha_n + \frac{1}{2} \| \sum_n y_n \alpha_n \boldsymbol{x}_n \|_2^2 - \sum_{m,n} \alpha_n \alpha_m y_m y_n \boldsymbol{x}_m^\top \boldsymbol{x}_n \\
&= \sum_n \alpha_n - \frac{1}{2} \sum_{m,n} \alpha_n \alpha_m y_m y_n \boldsymbol{x}_m^\top \boldsymbol{x}_n
\end{aligned}
$$

The dual problem

Maximizing the dual under the constraints

$$
\begin{aligned}
\max_{\boldsymbol{\alpha}} \quad & g(\{\alpha_n\}, \{\lambda_n\}) = \sum_n \alpha_n - \frac{1}{2} \sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{x}_m^\top \boldsymbol{x}_n \\
\text{s.t.} \quad & \alpha_n \geq 0, \quad \forall\ n \\
& \sum_n \alpha_n y_n = 0 \\
& C - \lambda_n - \alpha_n = 0, \quad \forall\ n \\
& \lambda_n \geq 0, \quad \forall\ n
\end{aligned}
$$

We can simplify as the objective function does not depend on λ_n. Specifically, we can combine the constraints involving λ_n resulting in the following inequality constraint: $\alpha_n \leq C$:

$$
\begin{aligned}
C - \lambda_n - \alpha_n = 0,\ \lambda_n \geq 0 \iff & \lambda_n = C - \alpha_n \geq 0 \\
\iff & \alpha_n \leq C
\end{aligned}
$$

Dual formulation of SVM

Dual is also a convex quadratic program

$$\max_{\boldsymbol{\alpha}} \quad \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{x}_m^\top \boldsymbol{x}_n$$

$$\text{s.t.} \quad 0 \leq \alpha_n \leq C, \quad \forall\ n$$

$$\sum_n \alpha_n y_n = 0$$

- There are N dual variables α_n, one for each data point
- Independent of the size d of $\mathbf{x}$: SVM scales better for high-dimensional feature.
- May seem like a lot of optimization variables when N is large, but many of the α_n's become zero. α_n is non-zero only if the n^{th} point is a support vector

Advantages of SVM

We've seen that the geometric formulation of SVM is equivalent to minimizing the empirical hinge loss. This explains why SVM:

1. Is less sensitive to outliers.
2. Maximizes distance of training data from the boundary
3. Generalizes well to many nonlinear models.
4. Only requires a subset of the training points.
5. Scales better with high-dimensional data.

The last thing left to consider is non-linear decision boundaries, or kernel SVMs

Kernel SVM

Non-linear basis functions in SVM

- What if the data is not linearly separable?
- We can transform the feature vector $\mathbf{x}$ using non-linear basis functions. For example,

$$\phi(\boldsymbol{x}) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1 x_2 \\ x_1^2 \\ x_2^2 \end{bmatrix}$$

- Replace $\mathbf{x}$ by $\phi(\mathbf{x})$ in both the primal and dual SVM formulations

Primal and Dual SVM Formulations: Kernel Versions

Primal

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n$$

$$\text{s.t.} \quad y_n[\boldsymbol{w}^\top \phi(\boldsymbol{x}_n) + b] \geq 1 - \xi_n, \quad \forall \ n$$

$$\xi_n \geq 0, \quad \forall \ n$$

Dual

$$\max_{\boldsymbol{\alpha}} \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n \phi(\boldsymbol{x}_m)^\top \phi(\boldsymbol{x}_n)$$

$$\text{s.t.} \quad 0 \leq \alpha_n \leq C, \quad \forall \ n$$

$$\sum_n \alpha_n y_n = 0$$

IMPORTANT POINT: In the dual problem, we only need $\phi(\boldsymbol{x}_m)^\top \phi(\boldsymbol{x}_n)$.

Dual Kernel SVM

We replace the inner products $\phi(\boldsymbol{x}_m)^\top \phi(\boldsymbol{x}_n)$ with a kernel function

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_n \alpha_n - \frac{1}{2} \sum_{m,n} y_m y_n \alpha_m \alpha_n k(\boldsymbol{x}_m, \boldsymbol{x}_n) \\ \text{s.t.} \quad & 0 \leq \alpha_n \leq C, \quad \forall\ n \\ & \sum_n \alpha_n y_n = 0 \end{aligned}$$

- $k(\mathbf{x}_m, \mathbf{x}_n)$ is a scalar and it is independent of the dimension of the feature vector $\phi(\mathbf{x})$.
- $k(\mathbf{x}_m, \mathbf{x}_n)$ roughly measures the similarity of $\mathbf{x}_m$ and $\mathbf{x}_n$.
- $k(\mathbf{x}_m, \mathbf{x}_n)$ is a kernel function if it is symmetric and positive-definite ($k(\mathbf{x}, \mathbf{x}) > 0$ for all $\mathbf{x} > 0$).

Examples of popular kernel functions

We do not need to know the exact form of $\phi(\boldsymbol{x})$, which lets us define much more flexible nonlinearities.

- Dot product:

$$k(\mathbf{x}_m, \mathbf{x}_n) = \mathbf{x}_m^\top \mathbf{x}_n$$

- Dot product with PD matrix $\mathbf{Q}$:

$$k(\mathbf{x}_m, \mathbf{x}_n) = \mathbf{x}_m^\top \mathbf{Q} \mathbf{x}_n$$

- Polynomial kernels:

$$k(\mathbf{x}_m, \mathbf{x}_n) = (1 + \mathbf{x}_m^\top \mathbf{x}_n)^d, \quad d \in \mathbb{Z}^+$$

- Radial basis function:

$$k(\mathbf{x}_m, \mathbf{x}_n) = \exp\left(-\gamma \left\|\mathbf{x}_m - \mathbf{x}_n\right\|^2\right) \text{ for some } \gamma > 0$$

and many more.

Test prediction

Learning $\boldsymbol{w}$ and b:

$$\boldsymbol{w} = \sum_n \alpha_n y_n \phi(\boldsymbol{x}_n)$$

$$b = y_n - \boldsymbol{w}^\top \phi(\boldsymbol{x}_n) = y_n - \sum_m \alpha_m y_m k(\boldsymbol{x}_m, \boldsymbol{x}_n)$$

But for test prediction on a new point $\mathbf{x}$, do we need the form of $\phi(\mathbf{x})$ in order to find the sign of $\boldsymbol{w}^\top \phi(\boldsymbol{x}) + b$? Fortunately, no!

Test Prediction:

$$h(\boldsymbol{x}) = \text{SIGN}(\sum_n y_n \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x}) + b)$$

At test time it suffices to know the kernel function! So we really do not need to know ϕ.

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n)$ for $n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

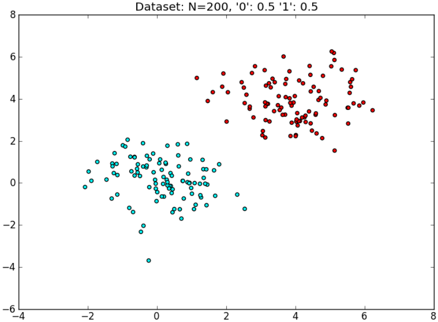


Image Source: `https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n) \text{ for } n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Here is the decision boundary with linear soft-margin SVM

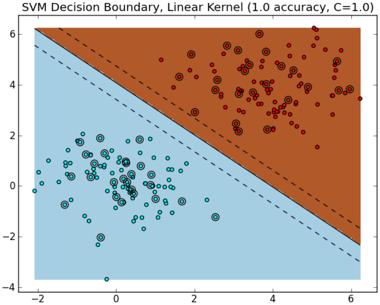


Image Source: `https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n)$ for $n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

What if the data is not linearly separable?

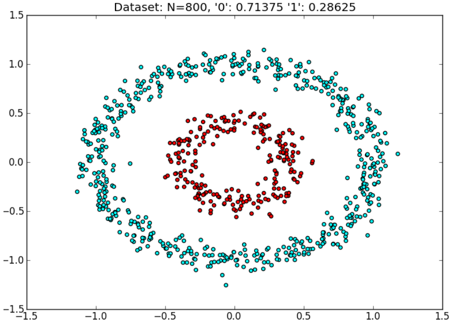


Image Source: `https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n)$ for $n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

The linear decision boundary is pretty bad

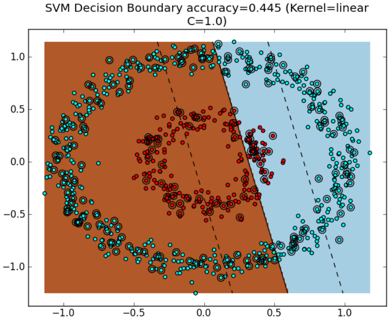
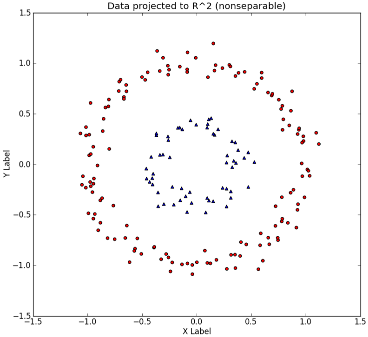


Image Source: `https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n)$ for $n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Use kernel $\phi(x) = [x_1, x_2, x_1^2 + x_2^2]$ to transform the data in a 3D space



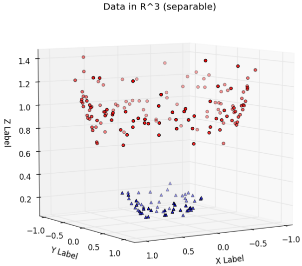


Image Source: `https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n) \text{ for } n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Then find the decision boundary. How? Solve the Dual problem

$$\max_{\boldsymbol{\alpha}} \sum_n \alpha_n - \frac{1}{2} \sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_m)^\top \boldsymbol{\phi}(\boldsymbol{x}_n)$$

$$\text{s.t.} \quad 0 \leq \alpha_n \leq C, \quad \forall\ n$$

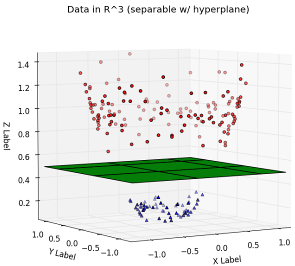
$$\sum_n \alpha_n y_n = 0$$

Then find $\mathbf{w}$ and b. Predict $y = \text{sign}(\mathbf{w}^T \phi(\mathbf{x}) + b)$.

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n)$ for $n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Here is the resulting decision boundary



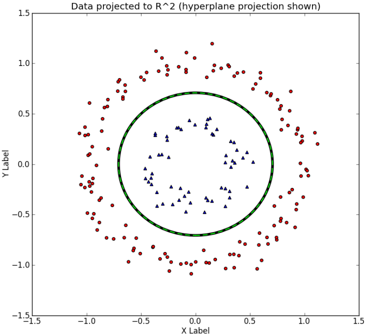


Image Source: `https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n)$ for $n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

In general, you don't need to concretely define $\phi(\mathbf{x})$. In the dual problem we can just use the kernel function $k(\mathbf{x}_m, \mathbf{x}_n)$. For cases where $\phi(\mathbf{x})$ is concretely defined, $k(\mathbf{x}_m, \mathbf{x}_n) = \phi(\mathbf{x}_m)^T \phi(\mathbf{x}_n)$.

$$
\begin{aligned}
\max_{\boldsymbol{\alpha}} & \sum_n \alpha_n - \frac{1}{2} \sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_m)^\top \boldsymbol{\phi}(\boldsymbol{x}_n) \\
\text{s.t.} \quad & 0 \leq \alpha_n \leq C, \quad \forall\ n \\
& \sum_n \alpha_n y_n = 0
\end{aligned}
$$

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n) \text{ for } n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

In general, you don't need to concretely define $\phi(\mathbf{x})$. In the dual problem we can just use the kernel function $k(\mathbf{x}_m, \mathbf{x}_n)$. For cases where $\phi(\mathbf{x})$ is concretely defined, $k(\mathbf{x}_m, \mathbf{x}_n) = \boldsymbol{\phi}(\mathbf{x}_m)^T \boldsymbol{\phi}(\mathbf{x}_n)$.

$$
\begin{aligned}
\max_{\boldsymbol{\alpha}} \sum_n \alpha_n - \frac{1}{2} \sum_{m,n} y_m y_n \alpha_m \alpha_n k(\mathbf{x}_m, \mathbf{y}_m) \\
\text{s.t.} \quad 0 \leq \alpha_n \leq C, \quad \forall\ n \\
\sum_n \alpha_n y_n = 0
\end{aligned}
$$

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n) \text{ for } n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Effect of the choice of kernel: Polynomial kernel (degree 4)

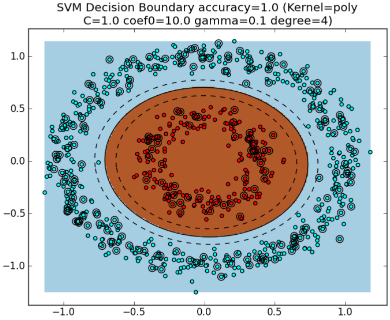


Image Source: `https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n)$ for $n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Effect of the choice of kernel: Radial Basis Kernel

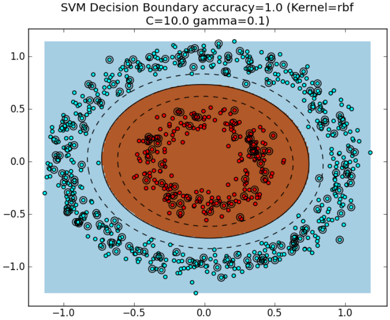


Image Source: `https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

Summary

You should know:

- Hinge loss function of SVM.
- How to derive the SVM dual.
- How to use the “kernel trick” in the dual SVM formulation to enable kernel SVM.
- How to compute an SVM prediction.