

Tufts Class #13:
Support Vector Machines (SVMs)
and Kernel Functions

Machine Learning (COMP 135): M. Allen, 02 Mar. 20

1

Data Separation

▶ Linear classification with a perceptron or logistic function look for a dividing line in the data (or a plane, or other linearly defined structure)

- ▶ Often *multiple* lines are possible
- ▶ Essentially, the algorithms are *indifferent*: they don't care which line we pick
- ▶ In the example seen here, either classification line separates data perfectly well

▶ Monday, 2 Mar. 2020 Machine Learning (COMP 135) 2

2

“Fragile” Separation

▶ As more data comes in, these classifiers may start to fail

- ▶ A separator that is too close to one cluster or the other now makes mistakes
- ▶ May happen even if new data follows same distribution seen in the training set

▶ Monday, 2 Mar. 2020 Machine Learning (COMP 135) 3

3

“Robust” Separation

▶ What we want is a **large margin** separator: a separation that has the largest distance possible from each part of our data-set

▶ This will often give much better performance when used on new data

▶ Monday, 2 Mar. 2020 Machine Learning (COMP 135) 4

4

Large Margin Separation

This is sometimes called the “widest road” approach

A **support vector machine (SVM)** is a technique that finds this road. The points that define the edges of the road are known as the **support vectors**.

- ▶ A new learning problem: **find** the separator with the largest margin
- ▶ This will be measured from the data points, on opposite sides, that are **closest together**

Monday, 2 Mar. 2020 Machine Learning (COMP 135) 5

5

Linear Classifiers and SVMs

Linear	
Weight equation	$\mathbf{w} \cdot \mathbf{x} = w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n$
Threshold function	$h_{\mathbf{w}} = \begin{cases} 1 & \mathbf{w} \cdot \mathbf{x} \geq 0 \\ 0 & \mathbf{w} \cdot \mathbf{x} < 0 \end{cases}$
SVM	
Weight equation	$\mathbf{w} \cdot \mathbf{x} + b = (w_1x_1 + w_2x_2 + \dots + w_nx_n) + b$
Threshold function	$h_{\mathbf{w}} = \begin{cases} +1 & \mathbf{w} \cdot \mathbf{x} \geq 0 \\ -1 & \mathbf{w} \cdot \mathbf{x} < 0 \end{cases}$

Monday, 2 Mar. 2020 Machine Learning (COMP 135) 6

6

Large Margin Separation

$\mathbf{w} \cdot \mathbf{x} + b = 0$

A key difference: the SVM is going to do this **without** learning and remembering weight vector \mathbf{w} . Instead, it will use features of the **data-items themselves**.

- ▶ Like a linear classifier; the SVM separates at the line where its learned vector of weights is zero

Monday, 2 Mar. 2020 Machine Learning (COMP 135) 7

7

Mathematics of SVMs

$\mathbf{w} \cdot \mathbf{x}^+ + b = +1$
 $\mathbf{w} \cdot \mathbf{x}^- + b = -1$

$\mathbf{w} \cdot (\mathbf{x}^+ - \mathbf{x}^-) = 2$
 $\frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot (\mathbf{x}^+ - \mathbf{x}^-) = \frac{2}{\|\mathbf{w}\|}$

$\|\mathbf{w}\| = \sqrt{w_1^2 + w_2^2 + \dots + w_n^2}$

- ▶ It turns out that the weight-vector \mathbf{w} for the largest margin separator has some important properties relative to the closest data-points on each side (\mathbf{x}^+ and \mathbf{x}^-)

Monday, 2 Mar. 2020 Machine Learning (COMP 135) 8

8

Mathematics of SVMs

▶ Through the magic of mathematics (Lagrangian multipliers, to be specific), we can derive a **quadratic programming** problem

1. We start with our data-set:

$$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\} \quad [\forall i, y_i \in \{+1, -1\}]$$

2. We then solve a **constrained optimization** problem:

$$W(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

$$\forall i, \alpha_i \geq 0$$

$$\sum_i \alpha_i y_i = 0$$

The goal: based on **known** values (\mathbf{x}_i, y_i) **find** the values we **don't know** (α_i) that:

1. Will **maximize** value of **margin** $W(\alpha)$
2. Satisfy the two numerical **constraints**

▶ Monday, 2 Mar. 2020

Machine Learning (COMP 135)

9

9

Mathematics of SVMs

▶ Although complex, a constrained optimization problem like this can be algorithmically solved to get the α_i values we want:

$$W(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

$$\forall i, \alpha_i \geq 0$$

$$\sum_i \alpha_i y_i = 0$$

A note about notation: these equations involve two different, necessary **products**:

1. The usual application of **weights** to **points**:

$$\mathbf{w} \cdot \mathbf{x}_i = w_1 x_{i,1} + w_2 x_{i,2} + \dots + w_n x_{i,n}$$

2. Products of **points** and **other points**:

$$\mathbf{x}_i \cdot \mathbf{x}_j = x_{i,1} x_{j,1} + x_{i,2} x_{j,2} + \dots + x_{i,n} x_{j,n}$$

▶ Once done, we can find the weight-vector and bias term if we want:

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \quad b = -\frac{1}{2} \left(\max_{i|y_i=-1} \mathbf{w} \cdot \mathbf{x}_i + \min_{j|y_j=+1} \mathbf{w} \cdot \mathbf{x}_j \right)$$

▶ Monday, 2 Mar. 2020

Machine Learning (COMP 135)

10

10

The Dual Formulation

▶ It turns out that we don't need to use the weights at all

▶ Instead, we can simply use the α_i values **directly**:

$$\mathbf{w} \cdot \mathbf{x}_i + b = \sum_j \alpha_j y_j (\mathbf{x}_i \cdot \mathbf{x}_j) + b$$

What we **usually** look for in a parametric method: the weights, \mathbf{w} , and offset, b , defining the classifier

What we can use **instead**: we compute an **equivalent** result based upon the α parameters, the outputs y , and products between data-points themselves (along with the standard offset)

▶ Monday, 2 Mar. 2020

Machine Learning (COMP 135)

11

11

The Dual Formulation

$$\mathbf{w} \cdot \mathbf{x}_i + b = \sum_j \alpha_j y_j (\mathbf{x}_i \cdot \mathbf{x}_j) + b$$

▶ Now, if we had to sum over **every** data-point as on the right-hand side of this equation, this would look very bad for a large data-set

▶ It turns out that these α_i values have a special property, however, that makes it feasible to use them as part of our classification function...

▶ Monday, 2 Mar. 2020

Machine Learning (COMP 135)

12

12

Sparseness of SVMs

So, when we do the calculation:

$$\sum_j \alpha_j y_j (\mathbf{x}_i \cdot \mathbf{x}_j) + b$$

We only have to sum over points \mathbf{x}_j that are in the set of **support vectors**, **ignoring** all others, since the related α values are all 0.

Thus, an SVM need only remember and use the values for a few support vectors, not those for all the rest of the data.

- ▶ The α_i values are 0 **everywhere except** at the support vectors (the points closest to the separator)

Monday, 2 Mar. 2020 Machine Learning (COMP 135) 13

13

Hard and Soft Margins

- ▶ We have slightly simplified one detail of how most SVMs actually work
- ▶ It is **not always true** that the support vectors lie on the margins, with nothing else in between them
- ▶ This is only true in the **hard margin case**
- ▶ SVMs can have **soft margins**, instead (and usually do) to deal with noisier data
- ▶ We weakened the requirement that no points lie between the margins
- ▶ **All points** within the margins then become the support vectors for classification

Monday, 2 Mar. 2020 Machine Learning (COMP 135) 14

14

Hard and Soft Margins

- ▶ Next lecture, we will see how to vary the margin strength in `sklearn`
- ▶ SVM models come with a regularization parameter (C , as in the case of classifiers) that can be:
 1. **Increased** to enforce a **harder** margin
 2. **Decreased** to allow a **softer** margin

Monday, 2 Mar. 2020 Machine Learning (COMP 135) 15

15

Another Nice Trick

$$\sum_j \alpha_j y_j (\mathbf{x}_i \cdot \mathbf{x}_j) + b$$

Using a **kernel** "trick", we can find a function that **transforms** the data into another form, where it is actually possible to separate it in a linear manner.

- ▶ The calculation uses dot-products of **data-points** with each other (instead of with weights)
- ▶ This will allow us to deal with data that is **not linearly separable**

Monday, 2 Mar. 2020 Machine Learning (COMP 135) 16

16

Transforming Non-Separable Data

A transformation function:
 $\varphi(\mathbf{x}) \quad \varphi : \mathbb{R}^n \rightarrow \mathbb{R}^m$
 maps data-vectors to new
 vectors, of either the same
 dimensionality ($m = n$) or a
 different one ($m \neq n$)

- ▶ If data that is not linearly separable, we can **transform** it
 - ▶ We **change** features used to represent our data
 - ▶ Really, we **don't care** what the data feature are, so long as we can get classification to work

Monday, 2 Mar. 2020 Machine Learning (COMP 135) 17

17

Transforming Non-Separable Data

$\varphi(\mathbf{x}) \quad \varphi : \mathbb{R}^n \rightarrow \mathbb{R}^m \quad \sum_j \alpha_j y_j (\varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)) + b$

Monday, 2 Mar. 2020 Machine Learning (COMP 135) 18

18

The "Kernel Trick"

Image source: Russel & Norvig, AI: A Modern Approach (Prentice Hall, 2010)

$$\varphi(x_1, x_2) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

Monday, 2 Mar. 2020 Machine Learning (COMP 135) 19

19

Simplifying the Transformation Function

- ▶ We can derive a simpler (2-dimensional) equation, **equivalent** to the cross-product needed when doing SVM computations in the transformed (3-dimensional) space:

$$\begin{aligned} \varphi(\mathbf{x}) \cdot \varphi(\mathbf{z}) &= (x_1^2, x_2^2, \sqrt{2}x_1x_2) \cdot (z_1^2, z_2^2, \sqrt{2}z_1z_2) \leftarrow \text{Needed} \\ &= x_1^2z_1^2 + x_2^2z_2^2 + \sqrt{2}x_1x_2\sqrt{2}z_1z_2 \\ &= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 \leftarrow \begin{array}{l} 10 \text{ multiplications} \\ 2 \text{ additions} \end{array} \\ &= (x_1z_1 + x_2z_2)^2 \leftarrow \begin{array}{l} 3 \text{ multiplications} \\ 1 \text{ addition} \end{array} \\ &= (\mathbf{x} \cdot \mathbf{z})^2 \leftarrow \begin{array}{l} \text{Used} \\ \text{instead} \end{array} \end{aligned}$$

Monday, 2 Mar. 2020 Machine Learning (COMP 135) 20

20

The Kernel Function

$$k(\mathbf{x}, \mathbf{z}) = \varphi(\mathbf{x}) \cdot \varphi(\mathbf{z}) = (\mathbf{x} \cdot \mathbf{z})^2$$

- ▶ This final function (right side) is what the SVM will actually use to compute dot-products in its equations
- ▶ This is called the **kernel function**
- ▶ To make SVMs really useful we look for a kernel that:
 1. Separates the data usefully
 2. Is relatively efficient to calculate

▶ Monday, 2 Mar. 2020

Machine Learning (COMP 135) 21

21

Another Reason to Use Kernel Functions

$$\mathbf{w} \cdot \mathbf{x}_i + b = \sum_j \alpha_j y_j (\varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)) + b$$

- ▶ The SVM formulation generally uses dot-products of data-points (perhaps run through some kernel) rather than the standard product of features and weights
- ▶ We have cases where the kernel-data approach is **possible**, but the weights-based one is **not**
 - ▶ Some useful kernels, that are easy to compute, correspond to weight-equations applied to **very high dimensional** transforms of the original data
 - ▶ In some common cases, equivalent weight-data vectors are **infinite-dimensional**, and simply cannot be used in computation

▶ Monday, 2 Mar. 2020

Machine Learning (COMP 135) 22

22

This Week & Next

- ▶ **Topics:** SVMs and kernels
 - ▶ Readings linked from class website schedule page.
- ▶ **Assignments:**
 - ▶ Project 01: due Monday, 09 March, 5:00 PM
 - ▶ Feature engineering and classification for image data
 - ▶ Midterm Exam: Wednesday, 11 March
 - ▶ Practice exam distributed by end of this week
 - ▶ Review session in class, Monday, 09 March
- ▶ **Office Hours:** 237 Halligan
 - ▶ Monday, 10:30 AM – Noon
 - ▶ Tuesday, 9:00 AM – 1:00 PM
 - ▶ TA hours can be found on class website

▶ Monday, 2 Mar. 2020

Machine Learning (COMP 135) 23

23