# CS 335: Kernel SVMs

Dan Sheldon

November 18, 2014

## First, a quick review

Hard-margin SVM

$$\min_{w,b} \ \frac{1}{2}||\mathbf{w}||^2$$
$$\text{subject to} \quad y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 \quad \text{for all } i$$

Functional margin at least one for all training examples

1D Exercise: which examples have functional margin $= 1$? $> 1$?

## Support Vectors

**Definition**: in a hard-margin SVM, a support vector is a training example with functional margin exactly equal to one.

## Visualization

MATLAB demo: hard-margin SVM (1)

Another interpretation of functional margin constraint:

$$y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1$$

All training examples at least one contour from decision boundary

MATLAB demo: hard-margin SVM w/ outlier (2)

## Soft-Margin SVM

$$\min_{w,b,\xi} \ \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m}\xi_i$$
$$\text{subject to} \quad y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 - \xi_i \quad \text{for all } i$$
$$\xi_i \geq 0, \ i = 1, \ldots, m$$

1D exercise

MATLAB demo (3). Exercise: is C increasing or decreasing?

## Support Vectors

**Definition**: a support vector is a training example with functional margin less than or equal to one.

(i.e., it falls on the wrong side of the 1-contour)

(definition works for hard or soft margin SVM)

## Kernel SVM Motivation

- But what we really want is a flexible non-linear classifer
  MATLAB demo (4)
- How can we get this with SVMs?
- Kernel trick!

## Kernel Trick Starting Point: Dual Optimization

Original SVM problem (hard-margin)

$$\boxed{\begin{aligned} &\min_{w,b}\ \frac{1}{2}||\mathbf{w}||^2 \\ &\text{s.t.}\quad y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)}+b)\geq 1 \quad \text{for all } i \end{aligned}}$$

. . . but this is not the version we usually solve.

## Kernel Trick Starting Point: Dual Optimization

After some fancy tricks, we instead solve this problem (the *Lagrangian dual*):

$$\boxed{\begin{aligned} &\max_{\boldsymbol{\alpha}} \sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i,j=1}^m y^{(i)}y^{(j)}\alpha_i\alpha_j(\mathbf{x}^{(i)})^T\mathbf{x}^{(j)} \\ &\text{s.t.}\quad \alpha_i \geq 0, \quad \text{for all } i, \\ &\qquad \sum_{i=1}^m \alpha_i y^{(i)} = 0 \end{aligned}}$$

Then recover $b$ (not shown) and $\mathbf{w}$ from the $\alpha$ variables:

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)}$$

## Aside: Support Vectors

Interpret this:

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)}$$

**Fact**: $\alpha_i > 0$ only if $\mathbf{x}^{(i)}$ is a support vector.

$\Rightarrow$ $\mathbf{w}$ is a linear combination of support vectors

## Dot Products

**Observation**: learning problem and prediction rule only depend on training examples through dot products

Learning problem:

$$\boxed{\begin{aligned} &\max_{\boldsymbol{\alpha}} \sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i,j=1}^m y^{(i)}y^{(j)}\alpha_i\alpha_j(\mathbf{x}^{(i)})^T\mathbf{x}^{(j)} \\ &\text{s.t.}\quad \alpha_i \geq 0, \quad \text{for all } i, \\ &\qquad \sum_{i=1}^m \alpha_i y^{(i)} = 0 \end{aligned}}$$

## Dot Products

**Observation**: learning problem and prediction rule only depend on training examples through dot products

Prediction for new $\mathbf{x}$:

$$h_{\mathbf{w},b}(\mathbf{x}) = b + \mathbf{w}^T\mathbf{x} = b + \sum_{i=1}^m \alpha_i y^{(i)}(\mathbf{x}^{(i)})^T\mathbf{x}$$

## Kernel Trick

Suppose you have a black box $K(\cdot, \cdot)$ to compute the dot product for *any two feature vectors* $\mathbf{x}$ and $\mathbf{z}$:

$$K(\mathbf{x}, \mathbf{z}) := \mathbf{x}^T \mathbf{z}$$

Thought experiment: I hold feature vectors in a box. You can ask me only for dot products.

Can you still solve the learning problem? Make predictions?

## Kernel Trick

$$\boxed{\begin{aligned} &\max_{\boldsymbol{\alpha}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \\ &\text{s.t.} \quad \alpha_i \geq 0, \quad \text{for all } i, \\ &\qquad \sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \end{aligned}}$$

$$h_{\mathbf{w},b}(\mathbf{x}) = b + \mathbf{w}^T \mathbf{x} = b + \sum_{i=1}^{m} \alpha_i y^{(i)} K(\mathbf{x}^{(i)}, \mathbf{x})$$

## Kernel Trick

This doesn't seem that special...

Real trick: fancy non-linear feature expansions in a computationally efficient way

## Feature Mapping

Let $\phi$ be a feature mapping from original features to expanded features.

E.g., $\phi : \mathbb{R}^n \to \mathbb{R}^{n^2}$:

$$\phi(\mathbf{x}) = \begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ x_2 x_1 \\ x_2 x_2 \end{bmatrix} \text{ (products of two original features)}$$

## Kernel

Given any feature mapping $\phi$, the kernel corresponding to $\phi$ is

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{z})$$

(map to higher dimensional space, then take dot product)

## Example: Polynomial Kernel

**Important trick**: we can often compute kernel without actually doing the expansion

$$K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T \mathbf{z})^2$$

**Claim**: this is the kernel corresponding to $\phi(\mathbf{x}) = \begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ x_2 x_1 \\ x_2 x_2 \end{bmatrix}$

Exercise: verify this on board

## More Polynomial Kernels

Claim: these two are equivalent

$$\phi(\mathbf{x}) = \begin{bmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ x_1x_1 \\ x_1x_2 \\ x_2x_1 \\ x_2x_2 \end{bmatrix} \qquad K(\mathbf{x},\mathbf{z}) = (\mathbf{x}^T\mathbf{z}+1)^2$$

- Complexity of computing $\phi(\mathbf{x})^T\phi(\mathbf{z})$?
- Complexity of computing $(\mathbf{x}^T\mathbf{z}+1)^2$?

## Polynomial Kernel: Significance

- Compute $\phi(\mathbf{x})^T\phi(\mathbf{z})$: $O(n^2)$
- Compute $(\mathbf{x}^T\mathbf{z}+1)^2$: $O(n)$

Implement a non-linear feature expansion with no extra computational cost

- Compute $\mathbf{x}^T\mathbf{z}$: $O(n)$

## Even More Polynomial Kernels

$$K(\mathbf{x},\mathbf{z}) = (\mathbf{x}^T\mathbf{z}+1)^d$$

Corresponds to $\phi$ that takes all products of up to $d$ original features

$O(n)$ time to compute kernel instead of $O(n^d)$

## Gaussian Kernel

$$K(\mathbf{x},\mathbf{z}) = \exp(-\gamma||\mathbf{x}-\mathbf{z}||^2)$$

- Highly flexible, non-linear kernel
- Corresponds to infinite dimensional $\phi$ (cannot implement feature mapping, but can still use kernel)

## Gaussian Kernel Interpretation

$$K(\mathbf{x},\mathbf{z}) = \exp(-\gamma||\mathbf{x}-\mathbf{z}||^2)$$

- Intepretation: similarity of $\mathbf{x}$ to $\mathbf{z}$
  - Picture on board
  - $\gamma$ controls how close $\mathbf{x}$ and $\mathbf{z}$ need to be to be similar

* MATLAB demo