Support Vector Machines & Kernels
Lecture 6

David Sontag
New York University

Slides adapted from Luke Zettlemoyer and Carlos Guestrin, and Vibhav Gogate

Dual SVM derivation (1) – the linearly separable case

Original optimization problem:

$$\text{minimize}_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}.\mathbf{w}$$
$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1, \ \ \forall j$$

Rewrite constraints

One Lagrange multiplier per example

Lagrangian:

$$L(\mathbf{w}, \alpha) = \frac{1}{2}\mathbf{w}.\mathbf{w} - \sum_j \alpha_j \left[\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j - 1\right]$$
$$\alpha_j \geq 0, \ \ \forall j$$

Our goal now is to solve: $\min_{\vec{w},b} \ \max_{\vec{\alpha}\geq 0} \ L(\vec{w}, \vec{\alpha})$

Dual SVM derivation (2) – the linearly separable case

(Primal) $$\min_{\vec{w},b} \max_{\vec{\alpha}\geq 0} \frac{1}{2}||\vec{w}||^2 - \sum_j \alpha_j \left[(\vec{w}\cdot\vec{x}_j + b)\, y_j - 1\right]$$

Swap min and max

(Dual) $$\max_{\vec{\alpha}\geq 0} \min_{\vec{w},b} \frac{1}{2}||\vec{w}||^2 - \sum_j \alpha_j \left[(\vec{w}\cdot\vec{x}_j + b)\, y_j - 1\right]$$

Slater's condition from convex optimization guarantees that these two optimization problems are equivalent!

Dual SVM derivation (3) – the linearly separable case

(Dual) $$\max_{\vec{\alpha}\geq 0} \min_{\vec{w},b} \frac{1}{2}||\vec{w}||^2 - \sum_j \alpha_j \left[(\vec{w}\cdot\vec{x}_j + b)\, y_j - 1\right]$$

Can solve for optimal **w**, b as function of α :

$$\frac{\partial L}{\partial w} = w - \sum_j \alpha_j y_j x_j \quad \rightarrow \quad \mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j$$

$$\frac{\partial L}{\partial b} = -\sum_j \alpha_j y_j \quad \rightarrow \quad \sum_j \alpha_j y_j = 0$$

Substituting these values back in (and simplifying), we obtain:

(Dual) $$\max_{\vec{\alpha}\geq 0,\, \sum_j \alpha_j y_j = 0} \sum_j \alpha_j - \frac{1}{2}\sum_{i,j} \underbrace{y_i y_j \alpha_i \alpha_j}_{\text{scalars}} \underbrace{(\vec{x}_i \cdot \vec{x}_j)}_{\text{dot product}}$$

Sums over all training examples

Dual SVM derivation (3) – the linearly separable case

(Dual) $$\max_{\vec{\alpha}\geq 0} \min_{\vec{w},b} \frac{1}{2}||\vec{w}||^2 - \sum_j \alpha_j \left[(\vec{w}\cdot\vec{x}_j + b)\, y_j - 1\right]$$

Can solve for optimal **w**, b as function of α:

$$\frac{\partial L}{\partial w} = w - \sum_j \alpha_j y_j x_j \quad \rightarrow \quad \mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j$$

$$\frac{\partial L}{\partial b} = -\sum_j \alpha_j y_j \quad \rightarrow \quad \sum_j \alpha_j y_j = 0$$

Substituting these values back in (and simplifying), we obtain:

(Dual) $$\max_{\vec{\alpha}\geq 0,\ \sum_j \alpha_j y_j = 0} \sum_j \alpha_j - \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j \left(\vec{x}_i \cdot \vec{x}_j\right)$$

So, in dual formulation we will solve for α directly!

- **w** and b are computed from α (if needed)

Dual SVM derivation (3) – the linearly separable case

Lagrangian:

$$L(\mathbf{w},\alpha) = \tfrac{1}{2}\mathbf{w}.\mathbf{w} - \sum_j \alpha_j \left[\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j - 1\right]$$

$$\alpha_j \geq 0, \ \ \forall j$$

$\alpha_j > 0$ for some *j* implies constraint is tight. We use this to obtain *b*:

$$y_j\left(\vec{w}\cdot\vec{x}_j + b\right) = 1 \qquad (1)$$

$$y_j y_j\left(\vec{w}\cdot\vec{x}_j + b\right) = y_j \qquad (2)$$

$$\left(\vec{w}\cdot\vec{x}_j + b\right) = y_j \qquad (3)$$

$$\boxed{\begin{array}{l} \mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \\ b = y_k - \mathbf{w}.\mathbf{x}_k \\ \text{for any } k \text{ where } \alpha_k > 0 \end{array}}$$

Classification rule using dual solution

$$y \leftarrow \mathrm{sign}(\vec{w} \cdot \vec{x} + b)$$

$\Big\downarrow$ Using dual solution

$$y \leftarrow \mathrm{sign}\left[\sum_i \alpha_i y_i \underbrace{(\vec{x}_i \cdot \vec{x})} + b\right]$$

dot product of feature vectors of new example with support vectors

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$b = y_k - \mathbf{w}.\mathbf{x}_k$$

for any k where $C > \alpha_k > 0$

Dual for the non-separable case

Primal:

$$\text{minimize}_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_j \xi_j$$
$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j, \ \forall j$$
$$\xi_j \geq 0, \ \forall j$$

Solve for w,b, α:

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$
$$b = y_k - \mathbf{w}.\mathbf{x}_k$$

for any k where $C > \alpha_k > 0$

Dual:

$$\text{maximize}_{\alpha} \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j$$

$$\sum_i \alpha_i y_i = 0$$
$$C \geq \alpha_i \geq 0$$

What changed?

- Added upper bound of C on α_i!
- Intuitive explanation:
  - Without slack, $\alpha_i \rightarrow \infty$ when constraints are violated (points misclassified)
  - Upper bound of C limits the α_i, so misclassifications are allowed

Support vectors

- **Complementary slackness** conditions:

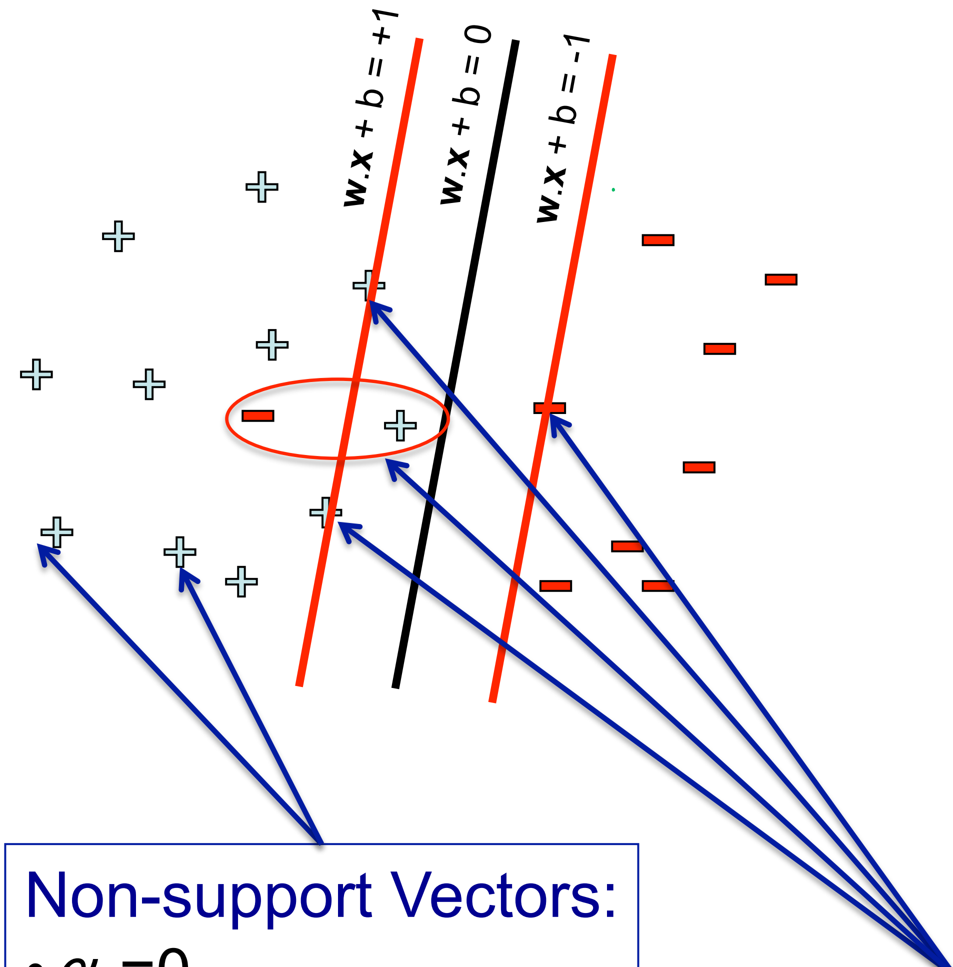
$$\alpha_j^* \left[y_j(\vec{w}^* \cdot \vec{x}_j + b) - 1 + \xi_j \right] = 0 \implies \alpha_j^* = 0 \;\vee\; y_j(\vec{w}^* \cdot \vec{x}_j + b) = 1 - \xi_j$$

$$\implies \alpha_j^* = 0 \;\vee\; y_j(\vec{w}^* \cdot \vec{x}_j + b) \leq 1$$

- **Support vectors**: points x_j such that $y_j(\vec{w}^* \cdot \vec{x}_j + b) \leq 1$ (includes all j such that $\alpha_j^* > 0$, but also additional points where $\alpha_j^* = 0 \wedge y_j(\vec{w}^* \cdot \vec{x}_j + b) \leq 1$)

- Note: the SVM dual solution may not be unique!

Dual SVM interpretation: Sparsity



$$\mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j$$

Final solution tends to be sparse

- α_j=0 for most j

- don’t need to store these points to compute w or make predictions

Non-support Vectors:
- α_j=0
- moving them will not change w

Support Vectors:
- $\alpha_j \geq 0$

SVM with kernels

$$\text{maximize}_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

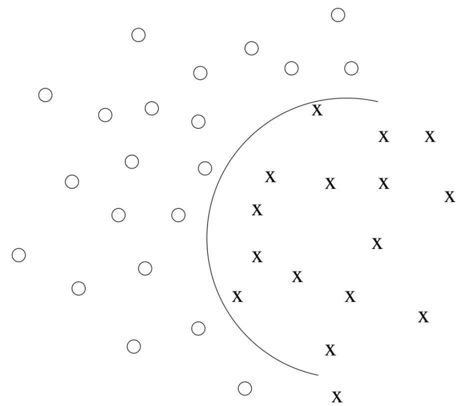
$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$
$$\sum_i \alpha_i y_i = 0$$
$$C \geq \alpha_i \geq 0$$

- Never compute features explicitly!!!
  - Compute dot products in closed form
- $O(n^2)$ time in size of dataset to compute objective
  - much work on speeding up

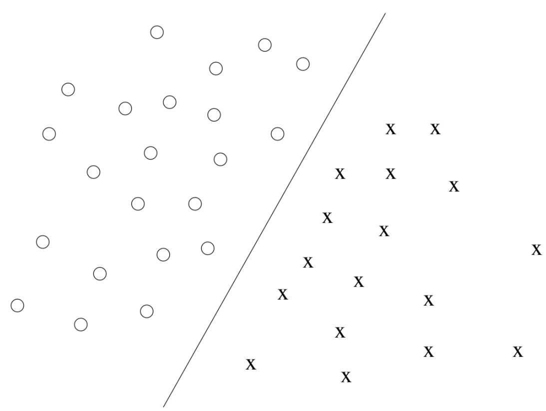
Predict with:

$$y \leftarrow \text{sign}\left[\sum_i \alpha_i y_i K(x_i, x) + b\right]$$

Quadratic kernel



Non-linear separator in the original $\mathbf{x}$-space



Linear separator in the feature ϕ-space

[Tommi Jaakkola]

Quadratic kernel

$$\begin{aligned} k(\mathbf{x},\mathbf{z}) &= (\mathbf{x}^T\mathbf{z}+c)^2 = \left(\sum_{j=1}^n x^{(j)}z^{(j)}+c\right)\left(\sum_{\ell=1}^n x^{(\ell)}z^{(\ell)}+c\right) \\ &= \sum_{j=1}^n\sum_{\ell=1}^n x^{(j)}x^{(\ell)}z^{(j)}z^{(\ell)} + 2c\sum_{j=1}^n x^{(j)}z^{(j)} + c^2 \\ &= \sum_{j,\ell=1}^n (x^{(j)}x^{(\ell)})(z^{(j)}z^{(\ell)}) + \sum_{j=1}^n(\sqrt{2c}x^{(j)})(\sqrt{2c}z^{(j)}) + c^2, \end{aligned}$$

Feature mapping given by:

$$\mathbf{\Phi}(\mathbf{x}) = [x^{(1)2}, x^{(1)}x^{(2)}, ..., x^{(3)2}, \sqrt{2c}x^{(1)}, \sqrt{2c}x^{(2)}, \sqrt{2c}x^{(3)}, c]$$

[Cynthia Rudin]

Common kernels

- Polynomials of degree exactly *d*

$$K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v})^d$$

- Polynomials of degree up to *d*

$$K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^d$$

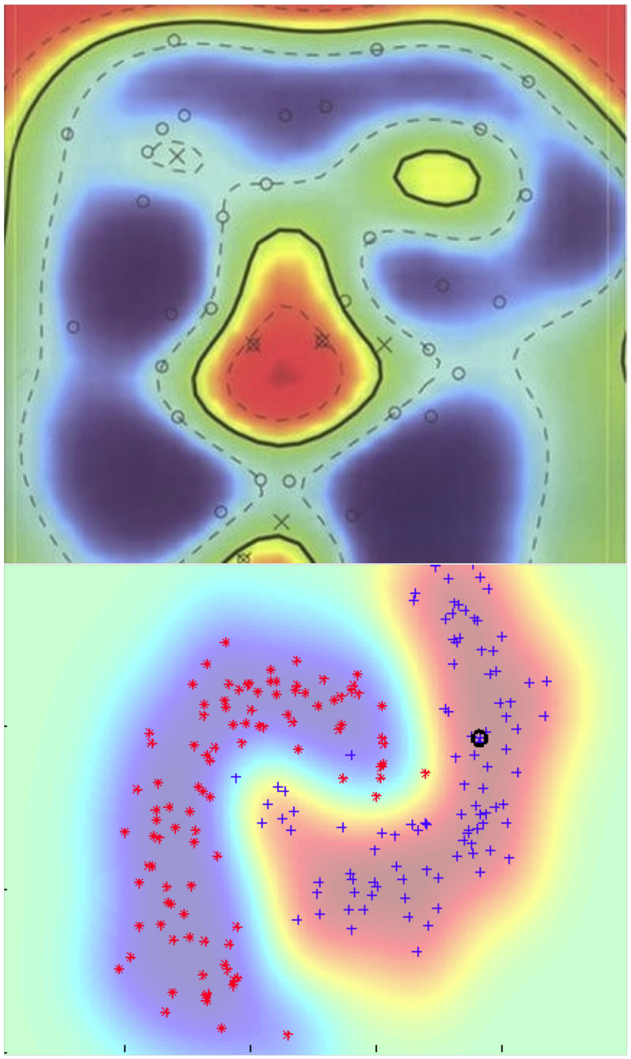
- Gaussian kernels

$$K(\vec{u}, \vec{v}) = \exp\left(-\frac{||\vec{u} - \vec{v}||_2^2}{2\sigma^2}\right)$$

Euclidean distance, squared

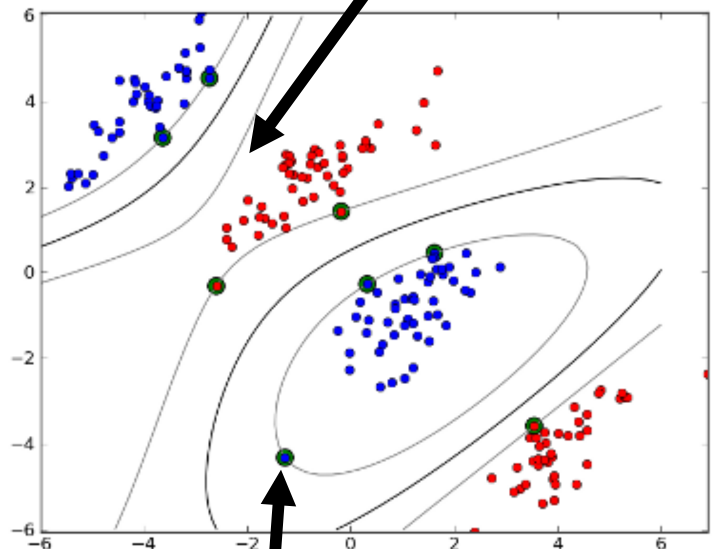
- And many others: very active area of research! (e.g., structured kernels that use dynamic programming to evaluate, string kernels, …)

Gaussian kernel



[Cynthia Rudin]

Level sets, i.e. w.x=r for some r



Support vectors

[mblondel.org]

Kernel algebra

kernel composition	feature composition
a) $k(\mathbf{x},\mathbf{v}) = k_a(\mathbf{x},\mathbf{v}) + k_b(\mathbf{x},\mathbf{v})$	$\boldsymbol{\phi}(\mathbf{x}) = (\boldsymbol{\phi}_a(\mathbf{x}), \boldsymbol{\phi}_b(\mathbf{x})),$
b) $k(\mathbf{x},\mathbf{v}) = f k_a(\mathbf{x},\mathbf{v}),\ f > 0$	$\boldsymbol{\phi}(\mathbf{x}) = \sqrt{f}\boldsymbol{\phi}_a(\mathbf{x})$
c) $k(\mathbf{x},\mathbf{v}) = k_a(\mathbf{x},\mathbf{v}) k_b(\mathbf{x},\mathbf{v})$	$\phi_m(\mathbf{x}) = \phi_{ai}(\mathbf{x})\phi_{bj}(\mathbf{x})$
d) $k(\mathbf{x},\mathbf{v}) = \mathbf{x}^T A \mathbf{v}$, A positive semi-definite	$\boldsymbol{\phi}(\mathbf{x}) = L^T\mathbf{x}$, where $A = LL^T$.
e) $k(\mathbf{x},\mathbf{v}) = f(\mathbf{x}) f(\mathbf{v}) k_a(\mathbf{x},\mathbf{v})$	$\phi(\mathbf{x}) = f(\mathbf{x})\phi_a(\mathbf{x})$

Q: How would you prove that the "Gaussian kernel" is a valid kernel?
A: Expand the Euclidean norm as follows:

$$\exp\left(-\frac{||\vec{u}-\vec{v}||_2^2}{2\sigma^2}\right) = \exp\left(-\frac{||\vec{u}||_2^2}{2\sigma^2}\right)\exp\left(-\frac{||\vec{v}||_2^2}{2\sigma^2}\right)\exp\left(\frac{\vec{u}\cdot\vec{v}}{\sigma^2}\right)$$

Then, apply (e) from above

To see that this is a kernel, use the Taylor series expansion of the exponential, together with repeated application of (a), (b), and (c):

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

The feature mapping is infinite dimensional!

[Justin Domke]

Overfitting?

- Huge feature space with kernels: should we worry about overfitting?
  - SVM objective seeks a solution with large **margin**
    - Theory says that large margin leads to good generalization (we will see this in a couple of lectures)
  - But everything overfits sometimes!!!
  - Can control by:
    - Setting C
    - Choosing a better Kernel
    - Varying parameters of the Kernel (width of Gaussian, etc.)