Do not distribute course material

- http://cs229.stanford.edu/notes/cs229-notes3.pdf
- https://www.svm-tutorial.com/
- https://nlp.stanford.edu/IR-book/html/htmledition/support-vector-machines-the-linearly-separable-case-1.html
- Advanced: https://svmtutorial.online/download.php?file=SVM_tutorial.pdf

Lecture
Support Vector Machines

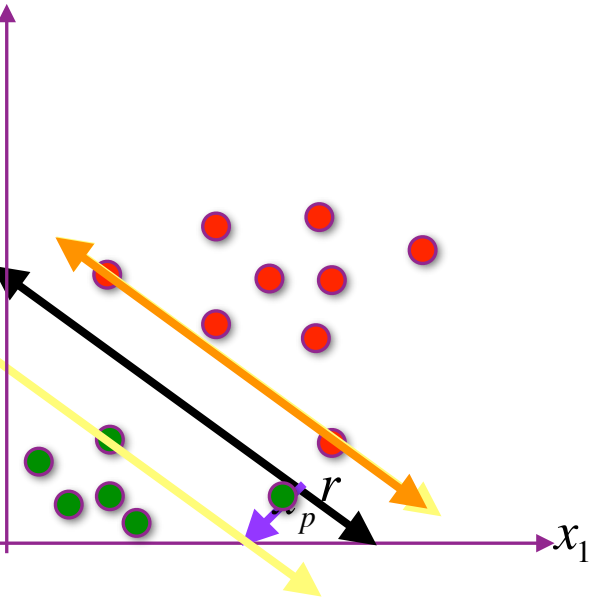
PROF. LINDA SELLIE

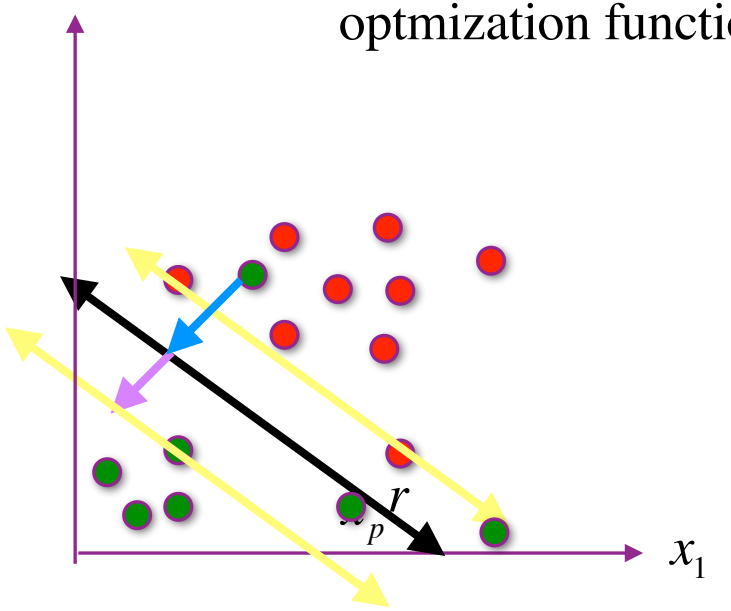
SOME SLIDES FROM PROF. RANGAN

Learning objectives:

- Understand the idea behind the geometric margin, functional margin, and canonical weights
- Create an objective function to find the hyperplane with the largest margin for linearly separable data
- Understand the hinge loss penalty
- Modify the objective function to allow for non-linearly separable data
- Understand the trade-off between the two terms in the soft margin objective function
- Know how to create a kernel function
- Understand the importance of the kernel function
- Know which vectors are support vectors

Soft-Margin SVM





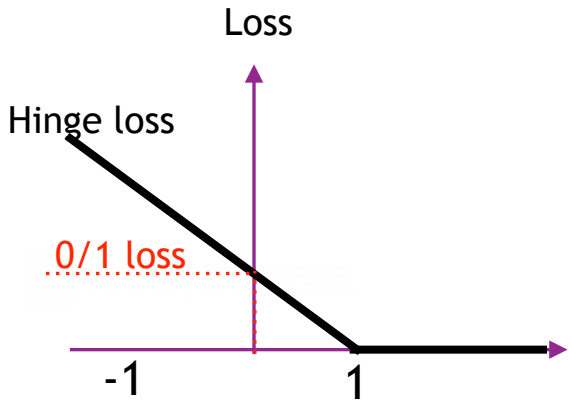
optmization function

$$\min_{w_0,\mathbf{w},\{\xi^{(i)}\}_{i=1}^{N}} \|\mathbf{w}\|_2^2 + C\sum_{i=1}^{N} \xi^{(i)}$$

$$\text{subject to } y^{(i)}(w_0 + \mathbf{w}^T\mathbf{x}^{(i)}) \geq 1 - \xi^{(i)} \quad \text{for all } i = 1,\ldots,N$$

$$\xi^{(i)} \geq 0$$

C is a tunable parameter. Gives relative importance of the error term



$$\xi^{(i)} = \begin{cases} 0 & y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0) \geq 1 \\ 1 - y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0) & otherwise \end{cases}$$

Outline

- Notation change, intuition, and finding how to compare hyperplanes - mathematically how do compare hyperplanes to find the one with the maximum margin. Can we turn this way of comparing hyperplanes into an objective function
- Support vector machines
  - ★ hard margin - find the constrained objective function when the data is linearly separable
  - ★ Dealing with non-linear data - "Soft" margins for SVM - New constrained objective function for the case where the data is not linearly separable
  - ★ Pegasos algorithm. Optimizer for soft margin SVM
  - ★ Dealing with non-linear data - feature transformation with the kernel trick - Show two popular feature maps

Simplifying our objective function

Rewriting our SVM objective function

$$\min_{w_0, \mathbf{w}, \{\xi^{(i)}\}_{i=1}^{N}} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{N} \xi^{(i)}$$

$$\text{subject to } y^{(i)}(w_0 + \mathbf{w}^T\mathbf{x}^{(i)}) \geq 1 - \xi^{(i)}$$

$$\xi^{(i)} \geq 0$$

Same as: $\xi^{(i)} \geq 1 - y^{(i)}\left(\mathbf{w}^T\mathbf{x}^{(i)} + w_0\right)$

Our SVM objective function with hinge loss:

Setting $\lambda = 1/C$

$$\min_{\mathbf{w}, w_0} \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \sum_{i=1}^{N} \max\left(0, 1 - y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0)\right)$$

regularizer

hinge Loss

Since $\xi^{(i)}$ is as small as possible

A balance between loss function and regularizer.

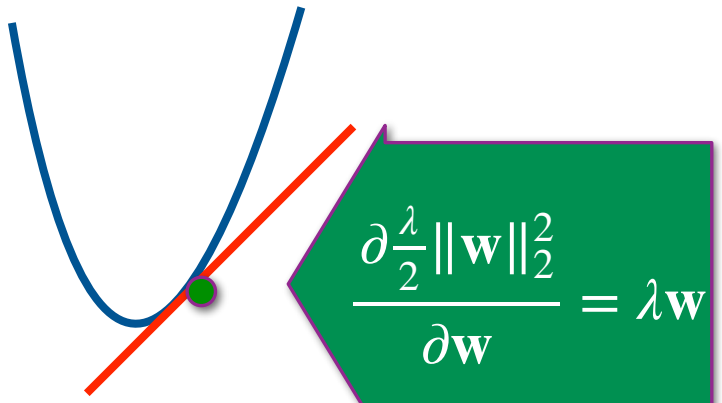
*In our optimizer, we will ignore the intercept term to make things easier

Our objective function is convex but not differentiable

We can use a sub-gradient. Derivation is beyond the scope of course.

Gradient

$$\frac{\lambda}{2}\|\mathbf{w}\|_2^2$$



$$\frac{\partial \frac{\lambda}{2}\|\mathbf{w}\|_2^2}{\partial \mathbf{w}} = \lambda \mathbf{w}$$

+

Sub-gradient of the hinge loss

$$\text{hinge loss} = \max\left(0, 1 - y^{(i)}\mathbf{w}^T\mathbf{x}\right)$$

$$\frac{\partial \text{hinge loss}}{\partial \mathbf{w}} = -y^{(i)}\mathbf{x}^{(i)}$$

?

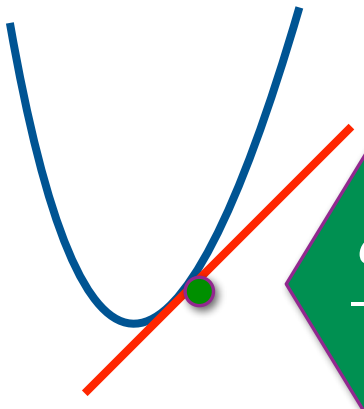
$$\frac{\partial \text{hinge loss}}{\partial \mathbf{w}} = 0$$

$$J(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \sum_{i=1}^{N} \max\left(0, 1 - y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0)\right)$$

$$\tilde{\nabla} J(\mathbf{w}) = \lambda\mathbf{w} + \begin{cases} 0 & \text{if } y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0) \geq 1 \\ -y^{(i)}\mathbf{x}^{(i)} & \text{otherwise} \end{cases}$$

Gradient

$$\frac{\lambda}{2}\|\mathbf{w}\|_2^2$$



$$\frac{\partial \frac{\lambda}{2}\|\mathbf{w}\|_2^2}{\partial \mathbf{w}} = \lambda \mathbf{w}$$

+

$$\frac{\partial \text{hinge loss}}{\partial \mathbf{w}} = -y^{(i)}\mathbf{x}^{(i)}$$

?

Sub-gradient: Linear global underestimate at this point

Sub-gradient of the hinge loss

$$\text{hinge loss} = \max\left(0, 1 - y^{(i)}\mathbf{w}^T\mathbf{x}\right)$$

$$J(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \sum_{i=1}^{N} \max\left(0, 1 - y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0)\right)$$

$$\tilde{\nabla} J(\mathbf{w}) = \lambda\mathbf{w} + \begin{cases} 0 & \text{if } y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0) \geq 1 \\ -y^{(i)}\mathbf{x}^{(i)} & \text{otherwise} \end{cases}$$

$$J(\mathbf{w}) = \underbrace{\frac{\lambda}{2}\|\mathbf{w}\|_2^2}_{\text{regularizer}} + \sum_{i=1}^{N} \underbrace{\max\left(0, 1 - y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)})\right)}_{\text{hinge Loss}}$$

$$\textbf{subgradient}(\mathbf{w}) = \begin{cases} \lambda\mathbf{w} & \textbf{if } y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)}) \geq 1 \\ \lambda\mathbf{w} - y^{(i)}\mathbf{x}^{(i)} & \textbf{otherwise} \end{cases}$$

"We did not incorporate a bias term in any of our experiments. We found that including an un-regularized bias term does not significantly change the predictive performance for any of the data sets used. Furthermore, most methods we compare to, including [21, 24, 37, 18], do not incorporate a bias term either. Nonetheless, there are clearly learning problems where the incorporation of the bias term could be beneficial." /https://www.cs.huji.ac.il/w~shais/papers/ShalevSiSrCo10.pdf

To keep it simple, we will not include a bias unit.

If N is large, batch gradient is slow

We will use stochastic sub-gradient descent with an adaptive learning rate

Stochastic Gradient Descent

$\mathbf{w}$ = random initialization

For t = 1,2,…,T:

Pick a random training example $(\mathbf{x}^{(i)}, y^{(i)})$

$$\# \hat{J}(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \max\left(0, 1 - y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} \quad)\right)$$

Use only one training example in objective function, N=1

$$\mathbf{w} = \mathbf{w} - \alpha\tilde{\nabla}\hat{J}(\mathbf{w})$$

The Pegasos Algorithm

$$\textbf{subgradient}(\mathbf{w}) = \begin{cases} \lambda \mathbf{w} & \textbf{if } y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)}) \geq 1 \\ \lambda \mathbf{w} - y^{(i)} \mathbf{x}^{(i)} & \textbf{otherwise} \end{cases}$$

$\mathbf{w}$ = random initialization

For t = 1,2,…,T:

Decrease the learning rate every iteration of the algorithm

Set $T = \tilde{O}\left(\frac{1}{\delta \lambda \epsilon}\right)$ err w* < w + ϵ with probability $1 - \delta$

Update the parameters by moving a small amount in the opposite direction of the sub gradient

Pair share: If α is small enough, will the function converge to a minimum value if enough iterations occur?

To keep it simple, we will not include a bias unit.

The Pegasos Algorithm

$$\textbf{subgradient}(\mathbf{w}) = \begin{cases} \lambda \mathbf{w} & \textbf{if } y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)}) \geq 1 \\ \lambda \mathbf{w} - y^{(i)} \mathbf{x}^{(i)} & \textbf{otherwise} \end{cases}$$

$\mathbf{w}$ = random initialization

For t = 1,2,…,T:

Pick a random training example $(\mathbf{x}^{(i)}, y^{(i)})$

$$\alpha = \frac{1}{\lambda \cdot t}$$

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

$\mathbf{w} = \mathbf{w} - \alpha\lambda\mathbf{w}$ # weight decay

(Shrink Weights)

else

$\mathbf{w} = \mathbf{w} - \alpha(\lambda\mathbf{w} - y^{(i)}\mathbf{x}^{(i)})$

(Shrink Weights; Move boundary)

Set $T = \tilde{O}\left(\frac{1}{\delta\lambda\epsilon}\right)$ err w* < w + ϵ with probability $1 - \delta$

Pair share: If α is small enough, will the function converge to a minimum value if enough iterations occur?

To keep it simple, we will not include a bias unit.

Modified Pegasos for Homework

$\mathbf{w}$ = 0, t = 0

For iter = 1,2,…,num_iters:

For j = 1, 2, …, N:

t=t+1

$$\alpha = \frac{1}{\lambda \cdot t}$$

if $y^{(j)}(\mathbf{w}^T \mathbf{x}^{(j)}) \geq 1$

$\mathbf{w} = \mathbf{w} - \alpha\lambda\mathbf{w}$ # weight decay

else

$\mathbf{w} = \mathbf{w} - \alpha(\lambda\mathbf{w} - y^{(j)}\mathbf{x}^{(j)})$

To keep it simple, we will not include a bias unit.

Outline

- Notation change, intuition, and finding how to compare hyperplanes - mathematically how do compare hyperplanes to find the one with the maximum margin. Can we turn this way of comparing hyperplanes into an objective function
- Support vector machines
  - ★ hard margin - find the constrained objective function when the data is linearly separable
  - ★ Dealing with non-linear data - "Soft" margins for SVM - New constrained objective function for the case where the data is not linearly separable
  - ★ Pegasos algorithm. Optimizer for soft margin SVM
  - ★ Dealing with non-linear data - feature transformation with the kernel trick - Show two popular feature maps

What if the data isn't linearly separable (part 2)?

SOME SLIDES WERE INSPIRED BY SLIDES FROM CMU 18-661

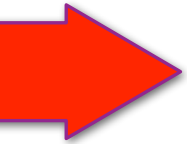
SOME SLIDES WERE INSPIRED BY SLIDES FROM VIVEK SRIKUMAR

Non-Linear Data

1. Soft margin
2. Transform features vector $\mathbf{x}$ into a new feature space $\Phi(\mathbf{x})$

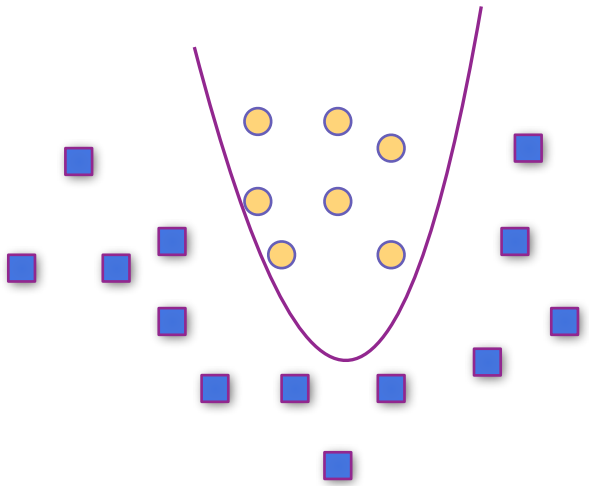
Pair share: What problems might arise if we transform our digit data using a degree 4 polynomial transformation, Φ_4?

SVM and feature transformation



- Feature transformation
- Support vectors
- The kernel trick
  (polynomial and RBF)

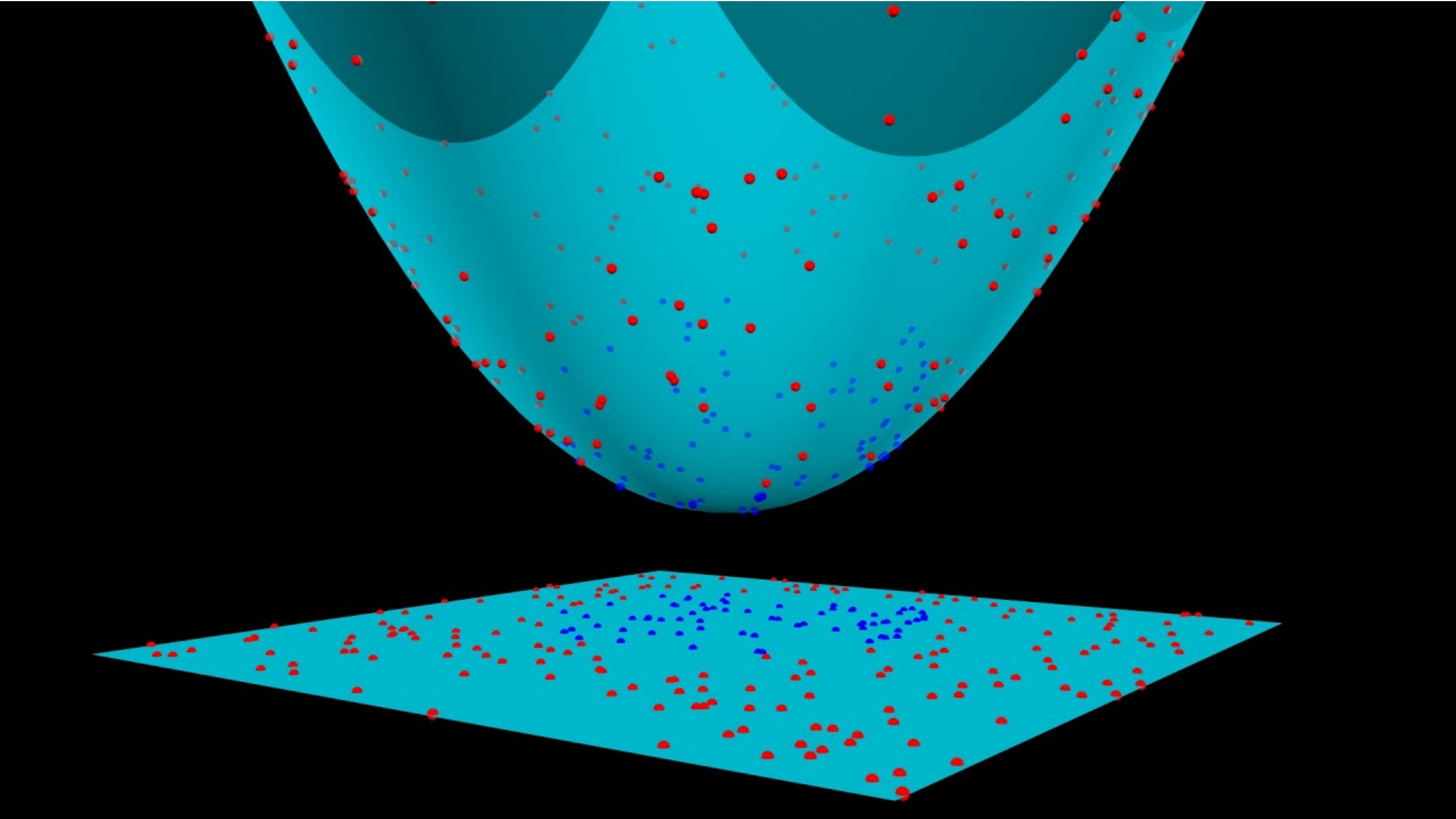
Non-Linear Data



$$\Phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1 \\ x_2 \\ x_1 x_2 \\ x_1^2 \\ x_2^2 \end{bmatrix}$$

Learn $\tilde{\mathbf{w}}$ in the transformed feature space (linear function)

Predict using $\text{sign}(\mathbf{w}^T \Phi(\mathbf{x}))$



high dimensional transformations

- What could be the problem?
  - The high computation costs
  - overfitting
- Kernel SVM is designed to deal (somewhat) with these issues.

How can we avoid paying the time/space cost of feature transformation?

Approach

1. Rewrite the algorithm so it only depends on inner products of feature vectors
2. Replace the inner product with the kernel function

You will not be tested on this

$$\Phi(\mathbf{x})^T\Phi(\mathbf{x}') \Rightarrow K(\mathbf{x}, \mathbf{x}')$$

The “Kernel Trick”

- For some transformations, we can compute

  $K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^T \Phi(\mathbf{x}')$

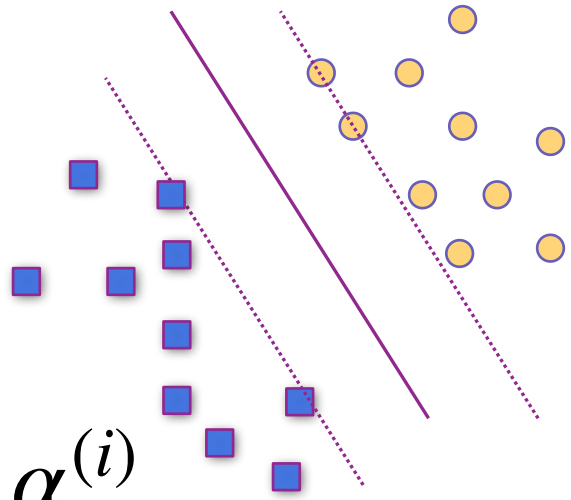
  without transforming the features. Instead, we perform operations in the original feature space to compute the value of $K(\mathbf{x}, \mathbf{x}')$.
- In class, we will discuss two different types of transformations:
  1. Polynomial transformation
  2. RBF

Hypothesis $\mathbf{w}$

SVM's are typically implemented using the **dual form.** The kernel trick is especially efficient in the dual form. The dual form is outside the scope of the class.

In this new approach $\mathbf{w}$ will be a linear combination of the training examples!

$$\mathbf{w} = \sum_{i=1}^{N} \alpha^{(i)} y^{(i)} \Phi(\mathbf{x}^{(i)})$$



We will keep track of a single variable per data point $\alpha^{(i)}$.

Predict for example $\mathbf{x}$: $\hat{y} = \text{sign}(\mathbf{w}^T \mathbf{x}) = \text{sign}\left(\left(\sum_{i=1}^{N} \alpha^{(i)} y^{(i)} \Phi(\mathbf{x}^{(i)})^T \Phi(\mathbf{x}) \right) + w_0 \right)$

$$\mathbf{w} = \sum_{i=1}^{N} \alpha^{(i)} y^{(i)} \Phi(\mathbf{x}^{(i)})$$

The steps to rewrite the algorithm are beyond the scope of the course.

The lecture slides presented in class have been moved to the end.

SVM and feature transformation

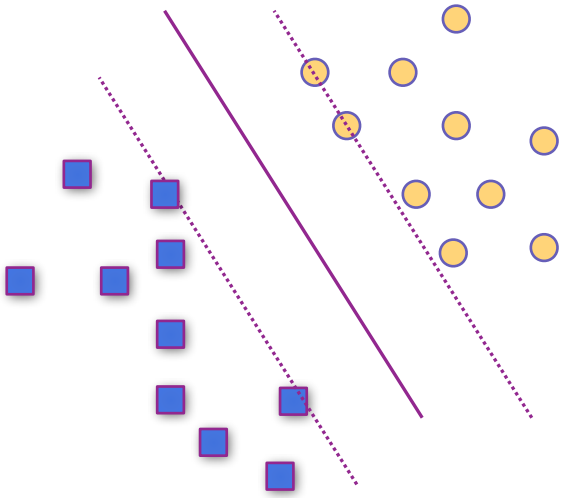
- Feature transformation
- Support vectors
- The kernel trick
  (polynomial and RBF)

SVM support vectors

The optimal $\mathbf{w}$ is a linear combination of the training examples whose $\alpha^{(i)} \neq 0$ (i.e. examples on the margin or inside the margin)

$$\mathbf{w} = \sum_{i=1}^{N} \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)}$$

$\alpha^{(i)} = 0$ if $y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + w_0) > 1$
All points correctly classified outside the margin



These examples are called the support vectors

Example (with rounded numbers….)

$$\mathbf{w} = \sum_{i=1}^{N} \alpha^{(i)} y^{(i)} \boldsymbol{x}^{(i)}$$

Given the set of training examples
((1,2.5), 1), ((0, 0.75), -1), ((1, 1), -1), ((2,2), 1), ((3,1), 1), ((2,3), 1), …

α^T = [0.9, 0, 1.4, 0, 0.5, 0, …, 0]

What is **w**?

$$\mathbf{w} = 0.9(1)\begin{bmatrix}1\\2.5\end{bmatrix} + 1.4(-1)\begin{bmatrix}1\\1\end{bmatrix} + 0.5(1)\begin{bmatrix}3\\1\end{bmatrix} = 0.9\begin{bmatrix}1\\2.5\end{bmatrix} + -1.4\begin{bmatrix}1\\1\end{bmatrix} + 0.5\begin{bmatrix}3\\1\end{bmatrix} = \begin{bmatrix}1\\1.35\end{bmatrix}$$

What is w_0?

No examples are inside the margin. Thus every support vector has a functional margin of one.

$$(1)\left(w_0 + \mathbf{w}^T\begin{bmatrix}3\\1\end{bmatrix}\right) = 1$$

$$w_0 + 4.35 = 1$$

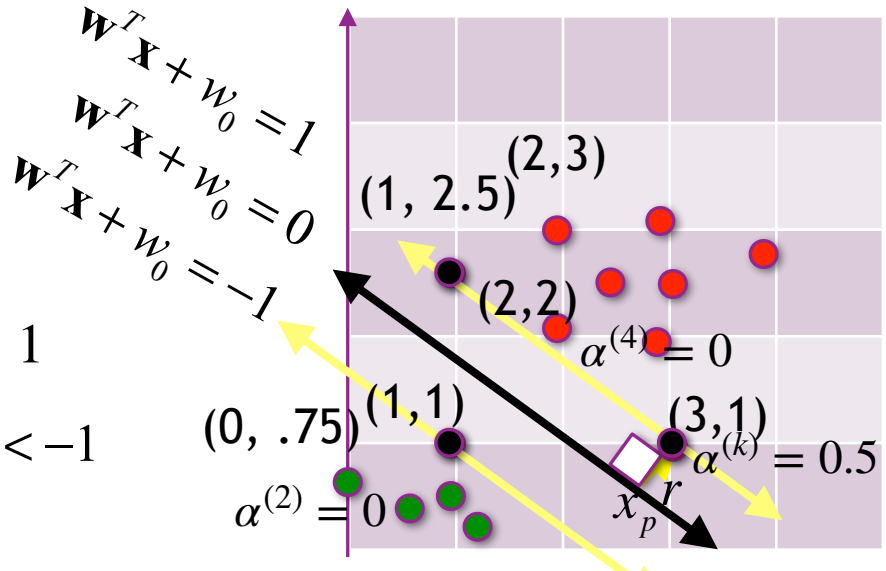
$$w_0 = -3.35$$

Note… I rounded these numbers, so… (If I hadn't rounded $w_2 = 1.33, w_0 = -3.33$)

$$\mathbf{w}^T(3,1)^T + w_0 = 1$$

$$\mathbf{w}^T(0,0.75)^T + w_0 < -1$$

$$\mathbf{w}^T(2,3)^T + w_0 > 1$$



Prediction of $\begin{bmatrix}1\\0\end{bmatrix}$

$$\mathbf{w} = \begin{bmatrix}1\\1.35\end{bmatrix} = 0.9\begin{bmatrix}1\\1.25\end{bmatrix} + -1.4\begin{bmatrix}1\\1\end{bmatrix} + 0.5\begin{bmatrix}3\\1\end{bmatrix}, \quad w_0 = -3.35$$

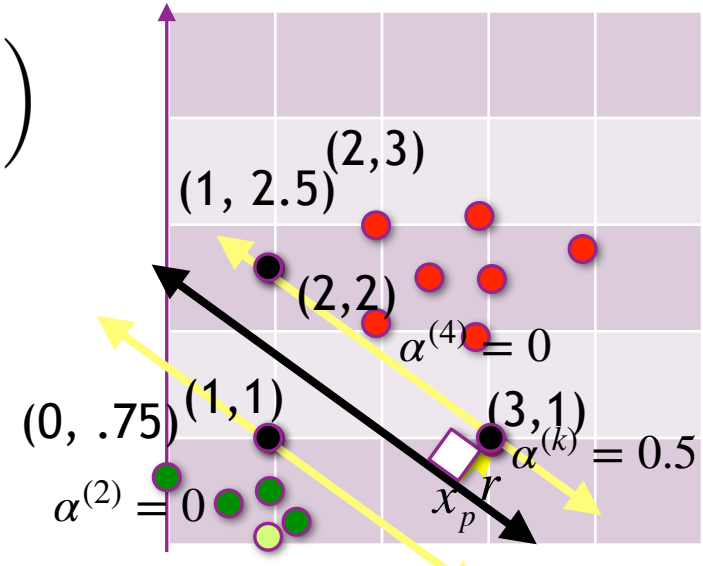
$$\text{sign}\left(\mathbf{w}^T\begin{bmatrix}1\\0\end{bmatrix} + w_0\right) = \text{sign}\left(\begin{bmatrix}1\\1.35\end{bmatrix}^T\begin{bmatrix}1\\0\end{bmatrix} - 3.35\right) = 1 - 3.35 = \text{sign}(-2.25) = -1$$

$$= \text{sign}\left(\left(0.9\begin{bmatrix}1\\1.25\end{bmatrix} + -1.4\begin{bmatrix}1\\1\end{bmatrix} + 0.5\begin{bmatrix}3\\1\end{bmatrix}\right)^T\begin{bmatrix}1\\0\end{bmatrix} - 3.35\right)$$

$$= \text{sign}\left(0.9K\left(\begin{bmatrix}1\\1.25\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}\right) + -1.4K\left(\begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}\right) + 0.5K\left(\begin{bmatrix}3\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}\right) - 3.35\right)$$

Linear kernel

$K(\mathbf{x}^{(1)}, \mathbf{x})$ $K(\mathbf{x}^{(3)}, \mathbf{x})$ $K(\mathbf{x}^{(5)}, \mathbf{x})$



SVM and feature transformation

- Feature transformation
- Support vectors
- The kernel trick
  (polynomial and RBF)

Polynomial kernel

$$\Phi_2(\mathbf{x})^T = [1, x_1, x_2, \ldots, x_d, x_1^2, x_2^2, \ldots, x_d^2, x_1x_2, \ldots, x_{d-1}x_d]$$

$$\Phi_2(\mathbf{x})^T\Phi_2(\mathbf{x}')$$

If d=200 then in z-space we have a feature vector of size approx 20,000

Kernel Motivation

$$g(\mathbf{x}) = \text{sign}\left(\sum_{i \in I} \alpha^{(i)} y^{(i)} (\mathbf{x}^{(i)T} \mathbf{x}) + w_0\right)$$

$$\mathbf{x} = [x_1, x_2]^T$$

$$\Phi_2 : R^2 \rightarrow R^6$$

$$\Phi_2(\mathbf{x}) = [1, x_1, x_2, x_1 x_2, x_1^2, x_2^2]^T$$

$$\Phi_2(-0.75, -0.25) = (1, -0.75, -0.25, (-0.75) \cdot (-0.25), (-0.75)^2, (-0.25)^2)$$

$$\Phi_2(-1,1) = (1, -1, 1, -1 \cdot 1, (-1)^2, 1^2)$$

$$\Phi_2(\mathbf{x})^T \Phi_2(\mathbf{x}') = [1, x_1, x_2, x_1 x_2, x_1^2, x_2^2] \cdot [1, x_1', x_2', x_1' x_2', x_1'^2, x_2'^2]$$

$$\Phi_2(\mathbf{x})^T \Phi_2(\mathbf{x}') = 1 + x_1 x_1' + x_2 x_2' + x_1 x_2 x_1' x_2' + x_1^2 x_1'^2 + x_2^2 x_2'^2$$

$$\Phi_2(-0.75, -0.25) \cdot \Phi_2(-1,1) = 1 + 0.75 - 0.25 + 0.75 \cdot (-0.25) + (-0.75)^2 + (-0.25)^2$$

Kernel motivation

$$g(\mathbf{x}) = \text{sign}\left(\sum_{i \in I} \alpha^{(i)} y^{(i)} (\mathbf{x}^{(i)T} \mathbf{x}) + w_0\right)$$

But if we use a different (but similar) transformation:

$$\Phi(\mathbf{x}) = [1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2]^T$$

$$\Phi(-0.75, -0.25) = \left(1, \sqrt{2} \cdot (-0.75), \sqrt{2} \cdot (-0.25), \sqrt{2}(-0.75) \cdot (-0.25), (-0.75)^2, (-0.25)^2\right)$$

$$\Phi(-1,1) = (1, -\sqrt{2}, \sqrt{2}, -\sqrt{2}, (-1)^2, 1^2)$$

$$K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}') = 1 + 2x_1x_1' + 2x_2x_2' + 2x_1x_2x_1'x_2' + x_1^2x_1'^2 + x_2^2x_2'^2$$

$$= (1 + \mathbf{x} \cdot \mathbf{x}')^2$$

$K((-1,1), (-0.75, -0.25)) =$ 2.25 ← Class exercise

$$= 1 + 2(0.75) + 2(-0.25) - 2(0.75)(0.25) + (-0.75)^2 + (-0.25)^2$$

$$= \left(1 + (0.75 - 0.25)\right)^2$$

So, instead of computing $\Phi(\mathbf{x})^T \Phi(\mathbf{x}')$, we compute $\mathbf{x} \cdot \mathbf{x}'$ in the original feature space, and then compute $(1 + \mathbf{x} \cdot \mathbf{x}')^2$

Learn more at: https://en.wikipedia.org/wiki/Polynomial_kernel

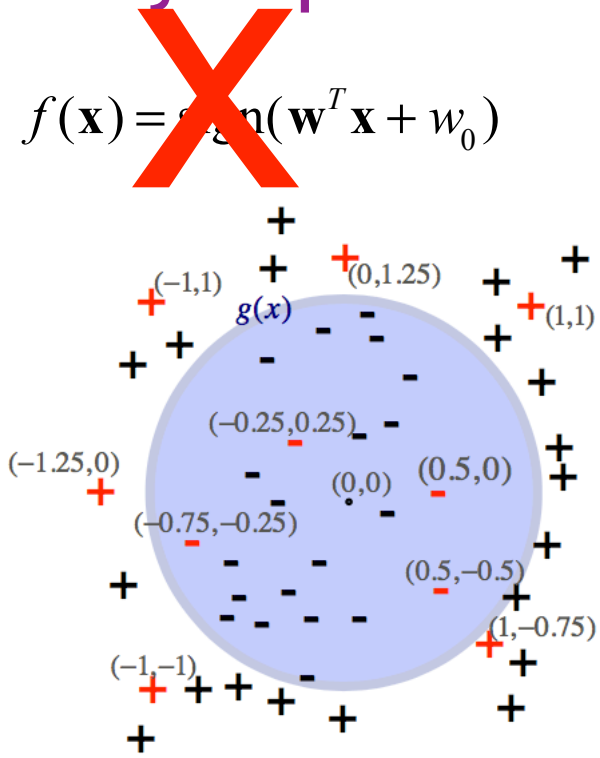
General form

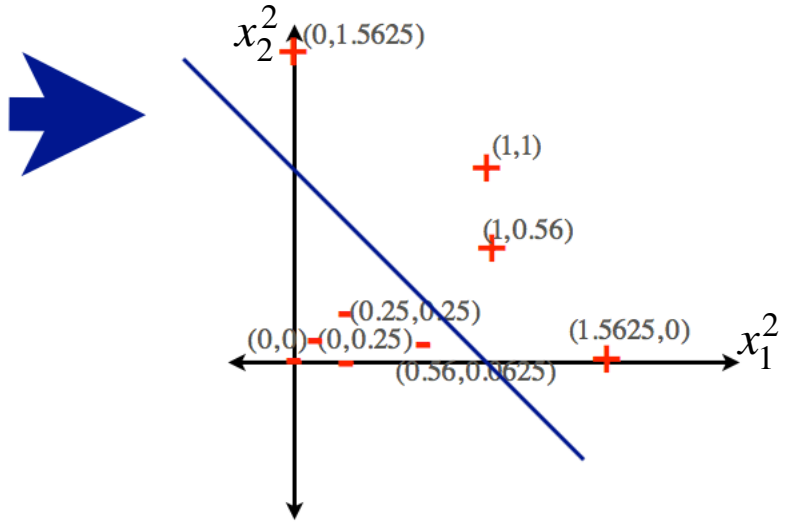
The polynomial kernel can be generalized to higher dimensions. A degree k polynomial is $K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x} \cdot \mathbf{x}')^k$

Example from the previous slide of $\Phi_2(\mathbf{x})$:
If d=200, then in z-space, we have a feature vector of size approx 20,000, but we do roughly the same amount of work to compute K(**x**,**x**') as if we hadn't transformed the features

Example

linearly separable?





If we transform the features using $\Phi(\mathbf{x}) = [1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2]^T$

$$\text{sign}\left(\left(\sum_{i\in I}\alpha^{(i)}y^{(i)}\Phi(\mathbf{x}^{(i)})\right)^T\Phi(\mathbf{x}) + w_0\right) \Rightarrow g(\mathbf{x}) = \text{sign}\left(\sum_{i\in I}\alpha^{(i)}y^{(i)}K(\mathbf{x}^{(i)},\mathbf{x}) + w_0\right)$$

$$K(\mathbf{x}^{(i)},\mathbf{x}) = (1 + \mathbf{x}^{(i)T}\mathbf{x})^2$$

The next slides contain material you will not be tested on.

Steps for rewriting the Pegasos algorithm:

1. Notice how $\mathbf{w}$ is updated during the algorithm. Create a new notation to capture how $\mathbf{w}$ is updated
2. Use the new notation to write $\mathbf{w}$ in terms of the training examples
3. Rewrite the Pegasos algorithm

The Pegasos Algorithm

Approach taken from Karl Stratos' notes on Pegasos

$\mathbf{w} = \mathbf{0}_d$ # column vector of d zeros

For t = 1,2,…,T:

Pick a random training example $(\mathbf{x}^{(i)}, y^{(i)})$

$$\alpha = \frac{1}{\lambda \cdot t}$$

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

$\mathbf{w} = \mathbf{w} - \alpha\lambda\mathbf{w}$ # weight decay

else

$\mathbf{w} = \mathbf{w} - \alpha(\lambda\mathbf{w} - y^{(i)}\mathbf{x}^{(i)})$

The Pegasos Algorithm

Approach taken from Karl Stratos' notes on Pegasos

$\mathbf{w} = \mathbf{0}_d$ # column vector of d zeros

For t = 1,2,…,T:

Pick a random training example $(\mathbf{x}^{(i)}, y^{(i)})$

$$\alpha = \frac{1}{\lambda \cdot t}$$

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

$\mathbf{w} = \mathbf{w} - \alpha\lambda\mathbf{w}$ # weight decay

else

~~$\mathbf{w} = \mathbf{w} - \alpha(\lambda\mathbf{w} - y^{(i)}\mathbf{x}^{(i)})$~~

$$\mathbf{w} = \mathbf{w} - \alpha\lambda\mathbf{w} + \alpha y^{(i)}\mathbf{x}^{(i)}$$

Shrink Weights

Move boundary

The Pegasos Algorithm

Approach taken from Karl Stratos' notes on Pegasos

$\mathbf{w} = \mathbf{0}_d$ # column vector of d zeros

For t = 1,2,…,T:

Pick a random training example $(\mathbf{x}^{(i)}, y^{(i)})$

$$\alpha = \frac{1}{\lambda \cdot t}$$

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

$\mathbf{w} = \mathbf{w} - \alpha\lambda\mathbf{w}$ # weight decay

else

$\mathbf{w} = \mathbf{w} - \alpha\lambda\mathbf{w} + \alpha y^{(i)}\mathbf{x}^{(i)}$

$$\mathbf{w} = \mathbf{w}\left(1 - \frac{1}{t}\right) + \alpha y^{(i)}\mathbf{x}^{(i)}$$

The Pegasos Algorithm

Approach taken from Karl Stratos' notes on Pegasos

$\mathbf{w} = \mathbf{0}_d$ # column vector of d zeros

For t = 1,2,…,T:

Pick a random training example $(\mathbf{x}^{(i)}, y^{(i)})$

$$\alpha = \frac{1}{\lambda \cdot t}$$

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

$\mathbf{w} = \mathbf{w} - \alpha\lambda\mathbf{w}$ # weight decay

else

$$\mathbf{w} = \mathbf{w}\left(1 - \frac{1}{t}\right) + \alpha y^{(i)}\mathbf{x}^{(i)}$$

$$\mathbf{w} = \mathbf{w}\left(1 - \frac{1}{t}\right) + \alpha v^{[t]}$$

The Pegasos Algorithm

Approach taken from Karl Stratos' notes on Pegasos

$\mathbf{w} = \mathbf{0}_d$ # column vector of d zeros

For t = 1,2,…,T:

Pick a random training example $(\mathbf{x}^{(i)}, y^{(i)})$

$$\alpha = \frac{1}{\lambda \cdot t}$$

$$\mathbf{w} = \mathbf{w}\left(1 - \frac{1}{t}\right) + \alpha v^{[t]}$$

The Pegasos Algorithm

Approach taken from Karl Stratos' notes on Pegasos

$$\mathbf{v}^{[t]} = \mathbf{1}\left[y^{(i)}(\mathbf{w}^{[t]T}\mathbf{x}^{(i)}) < 1\right] y^{(i)}\mathbf{x}^{(i)} \qquad \mathbf{1}\left[y(\mathbf{w}^{[t]T}\mathbf{x}) < 1\right] = \begin{cases} 1 & y(\mathbf{w}^{[t]T}\mathbf{x}) < 1 \\ 0 & \text{otherwise} \end{cases}$$

$\mathbf{w} = \mathbf{0}_d$ # column vector of d zeros

For t = 1,2,…,T:

Pick a random training example $(\mathbf{x}^{(i)}, y^{(i)})$

$$\alpha = \frac{1}{\lambda \cdot t}$$

$$\mathbf{w} = \mathbf{w}\left(1 - \frac{1}{t}\right) + \alpha v^{[t]}$$

$$\mathbf{w}^{[1]} = \mathbf{0}_d$$

$$\mathbf{w}^{[2]} = \frac{1}{\lambda}\mathbf{v}^{[1]} \qquad \alpha = \frac{1}{1 \cdot \lambda},\ \mathbf{w} = \mathbf{0}_d$$

$$\mathbf{w}^{[3]} = \frac{1}{2}\left(\frac{1}{\lambda}\mathbf{v}^{[1]}\right) + \frac{1}{2\lambda}\mathbf{v}^{[2]} \qquad \alpha = \frac{1}{2 \cdot \lambda}$$

The Pegasos Algorithm

Approach taken from Karl Stratos' notes on Pegasos

$$\mathbf{v}^{[t]} = \mathbf{1}\left[y^{(i)}(\mathbf{w}^{[t]T}\mathbf{x}^{(i)}) < 1\right] y^{(i)}\mathbf{x}^{(i)} \qquad \mathbf{1}\left[y(\mathbf{w}^{[t]T}\mathbf{x}) < 1\right] = \begin{cases} 1 & y(\mathbf{w}^{[t]T}\mathbf{x}) < 1 \\ 0 & \text{otherwise} \end{cases}$$

$\mathbf{w} = \mathbf{0}_d$ # column vector of d zeros

For t = 1,2,…,T:

Pick a random training example $(\mathbf{x}^{(i)}, y^{(i)})$

$$\alpha = \frac{1}{\lambda \cdot t}$$

$$\mathbf{w} = \mathbf{w}\left(1 - \frac{1}{t}\right) + \alpha v^{[t]}$$

$$\mathbf{w}^{[1]} = \mathbf{0}_d$$

$$\mathbf{w}^{[2]} = \frac{1}{\lambda}\mathbf{v}^{[1]} \qquad \alpha = \frac{1}{1 \cdot \lambda},\ \mathbf{w} = \mathbf{0}_d$$

$$\mathbf{w}^{[3]} = \frac{1}{2}\left(\frac{1}{\lambda}\mathbf{v}^{[1]}\right) + \frac{1}{2\lambda}\mathbf{v}^{[2]} = \frac{1}{2\lambda}\left(\mathbf{v}^{[1]} + \mathbf{v}^{[2]}\right)$$

$$\mathbf{w}^{[4]} = \frac{2}{3}\left(\frac{1}{2\lambda}(\mathbf{v}^{[1]} + \mathbf{v}^{[2]})\right) + \frac{1}{3\lambda}\mathbf{v}^{[3]} \qquad \alpha = \frac{1}{3 \cdot \lambda}$$

$$= \frac{1}{3\lambda}\left(\mathbf{v}^{[1]} + \mathbf{v}^{[2]} + + \mathbf{v}^{[3]}\right)$$

…

$$\mathbf{w}^{[t+1]} = \frac{1}{t\lambda}\sum_{\ell=1}^{t}\mathbf{v}^{[\ell]}$$

Writing w as a function of the training examples

New notation: if the k example is $(\mathbf{x}^{(i)}, y^{(i)})$ we will write $(\mathbf{x}^{(i_k)}, y^{(i_k)})$.

$$\mathbf{v}^{[k]} = \mathbf{1}\left[y^{(i_k)}(\mathbf{w}^{[k]T}\mathbf{x}^{(i_k)}) < 1\right] y^{(i_k)}\mathbf{x}^{(i_k)}$$

$$\mathbf{w}^{[t+1]} = \frac{1}{t\lambda}\sum_{k=1}^{t}\mathbf{v}^{[k]} = \frac{1}{\lambda t}\sum_{k=1}^{t} 1[y^{(i_k)}(\mathbf{w}^{[k]T}\mathbf{x}^{(i_k)}) < 1]y^{(i_k)}\mathbf{x}^{(i_k)}$$

Writing w as a function of the training examples continued

On this slide, we assume we have performed t updates to $\mathbf{w}$

$$\mathbf{w}^{[t+1]} = \frac{1}{\lambda t} \sum_{k=1}^{t} 1[y^{(i_k)}(\mathbf{w}^{[k]T}\mathbf{x}^{(i_k)}) < 1] y^{(i_k)}\mathbf{x}^{(i_k)}$$

For each $(\mathbf{x}^{(i)}, y^{(i)})$ we determine its contribution to $\mathbf{w}$

Define $\alpha^{(i)} = \frac{1}{t\lambda} \sum_{k\text{th example is } \mathbf{x}^{(i)},y^{(i)}} 1[y^{(i_k)}(\mathbf{w}^{[k]T}\mathbf{x}^{(i_k)}) < 1]$

Define $\alpha(i)$ = Number of times the margin is violated for ith example during training

Thus $\alpha^{(i)} = \frac{1}{\lambda t}\alpha(i)$

$$\mathbf{w}^{[t+1]} = \sum_{i=1}^{N} \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)}$$

During training, all we need to remember is how many times each training example violated the margin. We never need to compute $\mathbf{w}^{[t+1]}$ explicitly

The Kernalized Pegasos Algorithm

$$\mathbf{w}^{[t+1]} = \sum_{i=1}^{N} \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)} = \frac{1}{\lambda t} \sum_{i=1}^{N} \alpha(i) y^{(i)} \mathbf{x}^{(i)} \qquad \alpha^{(i)} = \frac{1}{\lambda t} \alpha(i)$$

$\mathbf{w} = \mathbf{0}_d$ # column vector of d zeros

For t = 1,2,…,T:

Pick a random training example $(\mathbf{x}^{(i)}, y^{(i)})$

$$\alpha = \frac{1}{\lambda \cdot t}$$

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

$\mathbf{w} = \mathbf{w} - \alpha\lambda\mathbf{w}$ # weight decay

else

$\mathbf{w} = \mathbf{w} - \alpha\lambda\mathbf{w} + \alpha y^{(i)} \mathbf{x}^{(i)}$

set $\alpha(i) = 0$ for $i \in 1,\ldots,N$

For t = 1,2,…,T:

Pick a random training example $(\mathbf{x}^{(i)}, y^{(i)})$

if $y^{(i)} \frac{1}{t\lambda} \overbrace{\left(\sum_{j=1}^{N} \alpha(j) y^{(j)} \mathbf{x}^{(j)T} \mathbf{x}^{(i)} \right)}^{\mathbf{w}^T} \geq 1$

No change

else

$\alpha(i) = \alpha(i) + 1$

The Kernalized Pegasos Algorithm

$$\mathbf{w}^{[t+1]} = \sum_{i=1}^{N} \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)} = \frac{1}{\lambda t} \sum_{i=1}^{N} \alpha(i) y^{(i)} \mathbf{x}^{(i)} \qquad \alpha^{(i)} = \frac{1}{\lambda t} \alpha(i)$$

$\mathbf{w} = \mathbf{0}_d$ # column vector of d zeros

For t = 1,2,…,T:

Pick a random training example $(\mathbf{x}^{(i)}, y^{(i)})$

$$\alpha = \frac{1}{\lambda \cdot t}$$

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

$\mathbf{w} = \mathbf{w} - \alpha\lambda\mathbf{w}$ # weight decay

else

$\mathbf{w} = \mathbf{w} - \alpha\lambda\mathbf{w} + \alpha y^{(i)} \mathbf{x}^{(i)}$

set $\alpha(i) = 0$ for $i \in 1,\ldots,N$

For t = 1,2,…,T:

Pick a random training example $(\mathbf{x}^{(i)}, y^{(i)})$

if $y^{(i)} \frac{1}{t\lambda} \left(\sum_{j=1}^{N} \alpha(j) y^{(j)} K(\mathbf{x}^{(j)}, \mathbf{x}^{(i)}) \right) \geq 1$

No change

else

$\alpha(i) = \alpha(i) + 1$

Prediction

After training, predicting the label of a new example $\mathbf{x}$:

$$\hat{y} = \text{sign}(\mathbf{w}^T\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{N} \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)T}\mathbf{x}\right)$$

$$= \text{sign}\left(\sum_{i=1}^{N} \alpha^{(i)} y^{(i)} K(\mathbf{x}^{(i)}, \mathbf{x})\right)$$

How did this save us any time?

Avoiding paying the time/space cost of feature transformation!

Key takeaway

If we can efficiently compute the inner product, we can *implicitly* work in high dimensional space

$$K(\mathbf{x}^{(i)}, \mathbf{x}) = \Phi(\mathbf{x}^{(i)})^T \Phi(\mathbf{x})$$

Next: The SVM Dual Form

THIS MATERIAL IS BEYOND THE SCOPE OF THE COURSE

Instead of modifying the Pegasos algorithm, we could have modified our objective function.

The next few slides show how this could be done.

This is a MUCH more common approach

Duality

Transforming our constrained objective function into a new objective function. (Thus we don't solve the original objective function)

- New objective function scales well with high-dimensional data
- Our hypothesis will be a function of the training examples (thus we can use the kernel function)

For SVM's the original constrained objective function (primal problem) and the dual formulation are equivalent problems. (This is not true in general when transforming a problem into its dual.)

Learning in a transformed feature space:

Pair share: if we use $\Phi(\mathbf{x})$, do we need to actually perform the transformation?

Primal

$$\min_{\mathbf{w},w_0} \|\mathbf{w}\|_2^2$$

subject to: $y^{(i)}(\mathbf{w}^T\Phi(\mathbf{x}^{(i)}) + w_0) \geq 1$ for all $i = 1 \ldots N$

Dual

$$\min_{\alpha} \sum_{i,j} \alpha^{(i)}\alpha^{(j)}y^{(i)}y^{(j)}\Phi(\mathbf{x}^{(i)})^T\Phi(\mathbf{x}^{(j)}) - \sum_i \alpha^{(i)}$$

$$\sum_{i=1}^{N} \alpha^{(i)}y^{(i)} = 0 \qquad \alpha^{(i)} \geq 0$$

We only need to know the result of $\Phi(\mathbf{x}^{(j)})^T\Phi(\mathbf{x}^{(i)})$

Prediction

$$g(\mathbf{x}) = \text{sign}\left(\mathbf{w}^T\Phi(\mathbf{x}) + w_0\right)$$

$$g(\mathbf{x}) = \text{sign}\left(\sum_{i\in I} \alpha^{(i)}y^{(i)}\Phi(\mathbf{x}^{(i)})^T\Phi(\mathbf{x}) + w_0\right)$$

We only need to know the result of $\Phi(\mathbf{x}^{(i)})^T\Phi(\mathbf{x})$

Support Vector $\mathbf{x}^{(i)}$

By computing the value of K without writing the transformation, we can have a computational advantage

Learning in a transformed feature space:

Pair share: if we use $\Phi(\mathbf{x})$, do we need to actually perform the transformation?

Primal

$$\min_{\mathbf{w},w_0} \|\mathbf{w}\|_2^2$$

subject to: $y^{(i)}(\mathbf{w}^T\Phi(\mathbf{x}^{(i)}) + w_0) \geq 1$ for all $i = 1 \ldots N$

Dual

$$\min_{\alpha} \sum_{i,j} \alpha^{(i)}\alpha^{(j)}y^{(i)}y^{(j)} K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) - \sum_i \alpha^{(i)}$$

$$\sum_{i=1}^{N} \alpha^{(i)} y^{(i)} = 0 \qquad \alpha^{(i)} \geq 0$$

We only need to know the result of $\Phi(\mathbf{x}^{(j)})^T\Phi(\mathbf{x}^{(i)})$

Prediction

$$g(\mathbf{x}) = \text{sign}\left(\mathbf{w}^T\Phi(\mathbf{x}) + w_0\right)$$

$$g(\mathbf{x}) = \text{sign}\left(\sum_{i \in I} \alpha^{(i)}y^{(i)} K(\mathbf{x}^{(i)}, \mathbf{x}) + w_0\right)$$

We only need to know the result of $\Phi(\mathbf{x}^{(i)})^T\Phi(\mathbf{x})$

Support Vector $\mathbf{x}^{(i)}$

By computing the value of K without writing the transformation, we can have a computational advantage

Learning in a transformed feature space:

Pair share: if we use $\Phi(\mathbf{x})$, do we need to actually perform the transformation?

Primal

$$\min_{\mathbf{w},w_0} \|\mathbf{w}\|_2^2$$

subject to: $y^{(i)}(\mathbf{w}^T\Phi(\mathbf{x}^{(i)}) + w_0) \geq 1$ for all $i = 1 \ldots N$

Dual

$$\min_{\alpha} \sum_{i,j} \alpha^{(i)}\alpha^{(j)}y^{(i)}y^{(j)} K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) - \ldots$$

$$\sum_{i=1}^{N} \alpha^{(i)} y^{(i)} = 0 \qquad \alpha^{(i)} \geq 0$$

Kernel trick: evaluate $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ without competing $\Phi(\mathbf{x}^{(j)})$ or $\Phi(\mathbf{x}^{(i)})$

We only need to know the result of $\Phi(\mathbf{x}^{(j)})^T\Phi(\mathbf{x}^{(i)})$

Prediction

$$g(\mathbf{x}) = \text{sign}\left(\mathbf{w}^T\Phi(\mathbf{x}) + w_0\right)$$

$$g(\mathbf{x}) = \text{sign}\left(\sum_{i \in I} \alpha^{(i)}y^{(i)} K(\mathbf{x}^{(i)}, \mathbf{x}) + w_0\right)$$

We only need to know the result of $\Phi(\mathbf{x}^{(i)})^T\Phi(\mathbf{x})$

Support Vector $\mathbf{x}^{(i)}$

By computing the value of K without writing the transformation, we can have a computational advantage

Lagrangian (hard margin case)

We will transform the hard margin case, the soft margin case is similar. We will assume the data is linearly separable

Primal SVM

- $\min_{\mathbf{w},w_0,\xi} \frac{1}{2}\|\mathbf{w}\|_2^2$

  Subject to $y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0) \geq 1$ for all $i = 1 \ldots N$

Lagrangian

- $L(\mathbf{w}, w_0, \alpha^{(i)}) = \frac{1}{2}\|\mathbf{w}\|_2^2 + \sum_{i=1}^{N} \alpha^{(i)} \left(1 - y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0)\right)$

  where $\alpha^{(i)} \geq 0$

Primal variables: $\mathbf{w}, w_0$
Dual variables: $\alpha^{(i)}, i \in \{1, \ldots, N\}$
(Lagrangian multipliers)

Dual formulation of SVM

$$\max_{\alpha^{(i)}, i \in \{1,\ldots,N\}} \sum_{i=1}^{N} \alpha^{(i)} - \frac{1}{2} \sum_{i,j} y^{(i)} y^{(j)} \alpha^{(i)} \alpha^{(j)} \mathbf{x}^{(i)T} \mathbf{x}^{(j)}$$

subject to $\sum_{i=1}^{N} \alpha^{(i)} y^{(i)} = 0$ and $\alpha^{(i)} \geq 0$

$$\mathbf{w} = \sum_{i=1}^{N} \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)}$$

Find $\alpha^{(j)} \neq 0$ solve

$$w_0 = y^{(j)} - \mathbf{w}^T \mathbf{x}^{(j)}$$

Steps:

* Minimize the Lagrangian function over the primal variables $\mathbf{w}, w_0$ by setting $\nabla_{\mathbf{w}} L = 0, \frac{\partial L}{\partial w_0} = 0.$
* Solve for primal variables wrt dual variables.
* Remove primal variables in the Lagrangian by substituting their equivalent dual variables

Lagrangian (hard margin case)

We will transform the hard margin case, the soft margin case is similar. We will assume the data is linearly separable

Primal SVM

- $\min_{\mathbf{w},w_0,\xi} \frac{1}{2}\|\mathbf{w}\|_2^2$

  Subject to $y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0) \geq 1$ for all $i = 1 \ldots N$

Lagrangian

- $L(\mathbf{w}, w_0, \alpha^{(i)}) = \frac{1}{2}\|\mathbf{w}\|_2^2 + \sum_{i=1}^{N} \alpha^{(i)} \left(1 - y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0)\right)$

  where $\alpha^{(i)} \geq 0$

Primal variables: $\mathbf{w}, w_0$
Dual variables: $\alpha^{(i)}, i \in \{1, \ldots, N\}$
(Lagrangian multipliers)

Dual formulation of SVM

$$\max_{\alpha^{(i)}, i \in \{1,\ldots,N\}} \sum_{i=1}^{N} \alpha^{(i)} - \frac{1}{2} \sum_{i,j} y^{(i)} y^{(j)} \alpha^{(i)} \alpha^{(j)} \mathbf{x}^{(i)T} \mathbf{x}^{(j)}$$

subject to $\sum_{i=1}^{N} \alpha^{(i)} y^{(i)} = 0$ and $\alpha^{(i)} \geq 0$

Steps:
* Minimize the Lagrangian function
ariables $\mathbf{w}, w_0$ by
), $\frac{\partial L}{\partial w_0} = 0.$
variables wrt dual

riables in the
ostituting their
ariables

Three key ideas from transformation you need to know (you don't need to know how the transformation works):

1. $\alpha^{(i)}(1 - y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0)) = 0$
2. $\mathbf{w} = \sum_{i=1}^{N} \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)}$
3. Find $\alpha^{(j)} \neq 0$ solve $w_0 = y^{(j)} - \mathbf{w}^T\mathbf{x}^{(j)}$

Example:

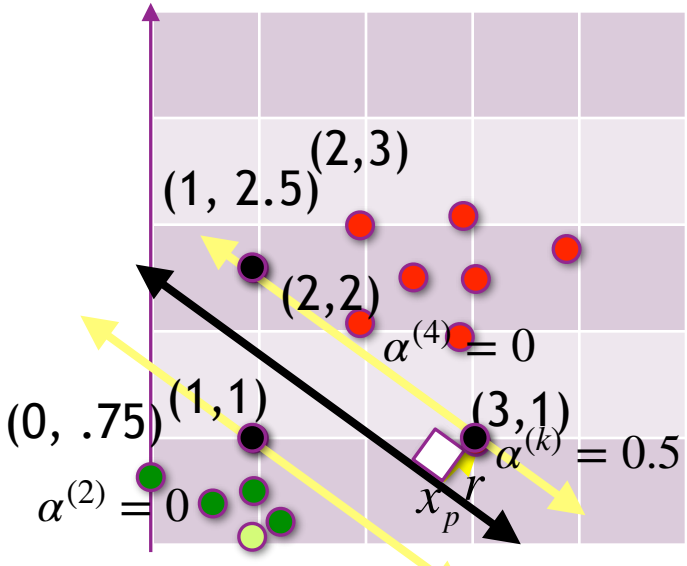
Given the set of training examples
((1,2.5), 1), ((0, 0.75), -1), ((1, 1), -1), ((2,2), 1), ((3,1), 1), ((2,3), 1), …

α^T = [0.9, 0, 1.4, 0, 0.5, 0, …, 0]

Any optimal solution must satisfy (stated without proof): $\alpha^{(i)}\left(1 - y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0)\right) = 0$

Thus $\alpha^{(i)} > 0$ and only if $1 - y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0) = 0$. $\mathbf{x}^{(i)}$ is on the margin!

Support vectors are $\mathbf{x}^{(i)}$ where $\alpha^{(i)} > 0$



Recovering $\mathbf{w}, w_0$

Dual formulation returns $\alpha = [\alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(N)}]^T$

Recovering **w**:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)} = \sum_{i \in I} \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)}$$

Only depends on support vectors! ($\alpha^{(i)} > 0$)

$I = \{i \mid \alpha^{(i)} \neq 0\}$

If $\alpha^{(i)} \neq 0$ then $x^{(i)}$ is a support vector,

The number of nonzero $\alpha^{(i)} \neq 0$ could be less than d

Recovering w_0: There exists some $\alpha^{(j)} \neq 0$

$$\alpha^{(j)} \underbrace{\left(y^{(j)} (w_0 + \mathbf{w}^T \mathbf{x}^{(j)}) - 1 \right)}_{0} = 0$$

Solve for w_0:

$$y^{(j)}(w_0 + \mathbf{w}^T \mathbf{x}^{(j)}) = 1$$

$$w_0 = y^{(j)} - \mathbf{w}^T \mathbf{x}^{(j)} = y^{(j)} - \sum_{i \in I} \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)} \mathbf{x}^{(j)}$$

Approach taken from Learning from Data

Creating the Dual SVM

- Steps to create the dual SVM
  - Create the Lagrangian
  - Minimize the Lagrangian over the primal variables by taking the gradient and setting the gradient to zero (thus we can write the primal variables in terms of the dual variables)
  - Substitute the primal variables with their equivalent dual formulas
  - Maximize the Lagrangian
  - Find $\mathbf{w}, w_0$ from dual variables

Lagrangian

Example: Given training examples (x^T,y):

((1, 2.5),1), ((2, 2),1), ((1,1),-1)

$$\min_{w_0,w_1,w_2} \frac{1}{2}[w_1, w_2]\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

Subject to:

$$y^{(1)}\left(w_0 + w_1 x_1^{(1)} + w_2 x_2^{(1)}\right) \geq 1,$$

$$y^{(2)}\left(w_0 + w_1 x_1^{(2)} + w_2 x_2^{(2)}\right) \geq 1,$$

$$y^{(3)}\left(w_0 + w_1 x_1^{(3)} + w_2 x_2^{(3)}\right) \geq 1$$

$$\min_{w_0,w_1,w_2} \frac{1}{2}[w_1, w_2]\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

Subject to:

$$\left(1-(w_0 + 1w_1 + 2.5w_2)\right) \leq 0$$

$$\left(1 - \left(w_0 + 2w_1 + 2w_2\right)\right) \leq 0$$

$$\left(1 - \left(-w_0 + -1w_1 + -1w_2\right)\right) \leq 0$$

The constrained quadratic optimization function can be rewritten as:

$$\min_{w_0,w_1,w_2} \max_{\alpha^{(1)}\geq 0} \max_{\alpha^{(2)}\geq 0} \max_{\alpha^{(3)}\geq 0} \frac{1}{2}[w_1, w_2]\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} + \alpha^{(1)}\left(1-(w_0 + 1w_1 + 2.5w_2)\right) + \alpha^{(2)}\left(1 - \left(w_0 + 2w_1 + 2w_2\right)\right)$$

$$+ \alpha^{(3)}\left(1 - \left(-w_0 + -1w_1 + -1w_2\right)\right)$$

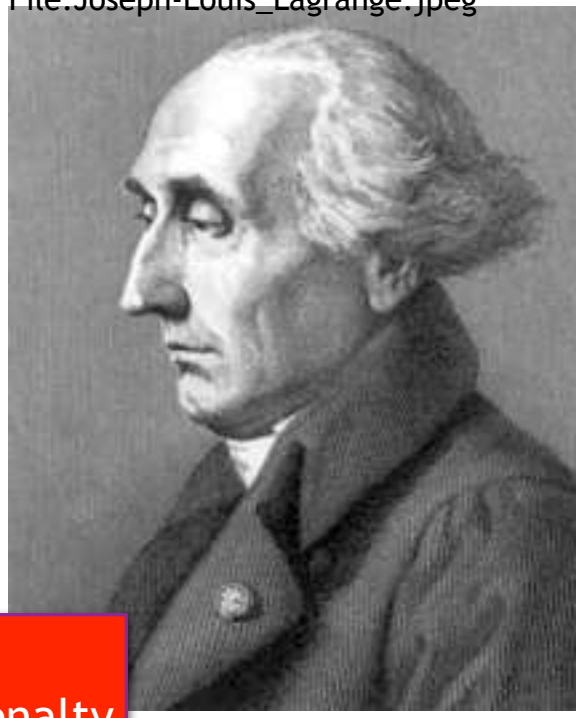
$$L(\mathbf{w}, w_0, \alpha) = \frac{1}{2}\|\mathbf{w}\|_2^2 + \sum_{i=1}^{N} \alpha^{(i)}\left(1 - y^{(i)}(w_0 + \mathbf{w}^T\mathbf{x}^{(i)})\right) \qquad L(\mathbf{w}, w_0, \alpha) = \frac{1}{2}\left\|\mathbf{w}\right\|^2 - \sum_{i=1}^{N} \alpha^{(i)}\left(y^{(i)}(w_0 + \mathbf{w}^T\mathbf{x}^{(i)}) - 1\right)$$

Background on solving constrained optimization

LAGRANGE DUALITY

Background: The Lagrangian

https://commons.wikimedia.org/wiki/File:Joseph-Louis_Lagrange.jpeg



Simplified problem

minimize **u**: $\frac{1}{2}\mathbf{u}^T\mathbf{u}$ ← primal problem

subject to: $\mathbf{a}^T\mathbf{u} \geq c$ where $\mathbf{a}^T\mathbf{u} \geq c$ is $a_1u_1 + a_2u_2 + \cdots + a_ru_r > c$

$\mathbf{a}'^T\mathbf{u} \geq c'$

If there is a *valid* solution, this is equal to: ← (c-aTu) <=0 otherwise it is ∞

minimize **u**: $\frac{1}{2}\mathbf{u}^T\mathbf{u} + \max_{\alpha \geq 0} \alpha(c - \mathbf{a}^T\mathbf{u}) + \max_{\alpha' \geq 0} \alpha'(c' - \mathbf{a}'^T\mathbf{u})$

← No constraint on **u**!

← But … complex 'Lagrangian' penalty

The Lagrangian function

$$L(\mathbf{u}, \alpha) = \frac{1}{2}\mathbf{u}^T\mathbf{u} + \alpha(c - \mathbf{a}^T\mathbf{u}) + \alpha'(c' - \mathbf{a}'^T\mathbf{u})$$

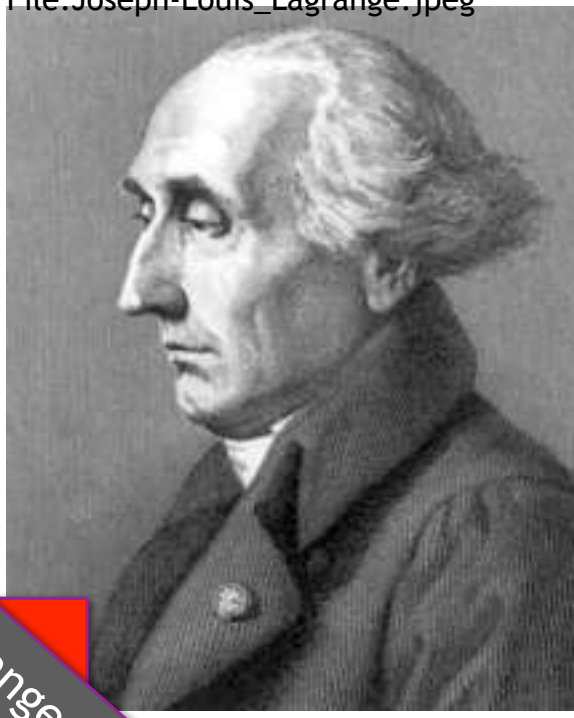
The optimization function

$$\min_{\mathbf{u}} \max_{\alpha \geq 0} L(\mathbf{u}, \alpha)$$

If there exists a solution to the primal problem which is a quadratic optimization with linear constraints there is ***strong duality:*** $\min_{\mathbf{u}} \max_{\alpha \geq 0} L(\mathbf{u}, \alpha) = \max_{\alpha \geq 0} \min_{\mathbf{u}} L(\mathbf{u}, \alpha)$

Background: The Lagrangian

https://commons.wikimedia.org/wiki/File:Joseph-Louis_Lagrange.jpeg



Simplified problem

minimize **u**: $\frac{1}{2}\mathbf{u}^T\mathbf{u}$

subject to: $\mathbf{a}^T\mathbf{u} \geq c$

$\mathbf{a}'^T\mathbf{u} \geq c'$

"However, if the primal problem is convex, i.e. if f and g_i are all convex functions and h_j are all affine functions, and if some additional conditions hold, then we do have strong duality. One such additional condition is Slater's condition, which requires that there exist a strictly feasible point x satisfying $g_i(x) < 0, i = 1,...,m$ and $h_j(x) = 0, j = 1,...,n$." from https://www.shivani-agarwal.net/Teaching/CIS-520/Spring-2018/Lectures/Reading/optimization.pdf

$+ \cdots + a_r u_r > c$

otherwise

nt on **u**!

First minimize with unconstrained **u** then maximize wrt α

Lagrange dual problem

If there is a *valid* solution, this

minimize **u**: $\frac{1}{2}\mathbf{u}^T\mathbf{u} + \max_{\alpha \geq 0} \alpha($

$+ \max_{\alpha' \geq 0} \alpha'($

The Lagrangian function

$$L(\mathbf{u}, \alpha) = \frac{1}{2}\mathbf{u}^T\mathbf{u} + \alpha(c - \mathbf{a}^T\mathbf{u}) + \alpha'(c' - \mathbf{a}'^T\mathbf{u})$$

The optimization function

$$\min_{\mathbf{u}} \max_{\alpha \geq 0} L(\mathbf{u}, \alpha)$$

If there exists a solution to the primal problem which is a quadratic optimization with linear constraints there is ***strong duality:*** $\min_{\mathbf{u}} \max_{\alpha \geq 0} L(\mathbf{u}, \alpha) = \max_{\alpha \geq 0} \min_{\mathbf{u}} L(\mathbf{u}, \alpha)$

SVM's can appear magical because we can transform the feature space $\Phi(\mathbf{x}^{(i)})$ (e.g. polynomial, RBF) and never have to compute $\Phi(\mathbf{x}^{(i)})$, instead we only need to compute the kernel $K(\mathbf{x}^{(i)},\mathbf{x}^{(j)}) = \Phi(\mathbf{x}^{(i)})^T \Phi(\mathbf{x}^{(j)})$ (In the description below we will not transform the feature space to keep the explanation clearer - but you can replace each $\mathbf{x}^{(i)}$ by $\Phi(\mathbf{x}^{(i)})$ yourself)

To show how this works, we transform our original problem into its duel formalization.

Given our constrained quadratic optimization problem: $\min \frac{1}{2}\|\mathbf{w}\|_2^2$ subject to $y^{(i)}(w_0 + \mathbf{w}^T\mathbf{x}^{(i)}) \geq 1$ for all i=1,...,N

We create the Lagrangian (each constraint is multiplied by a Lagrangian multiplier $\alpha^{(i)}$, for i = 1,...,N)

$$L(\mathbf{w}, w_0, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N}\alpha^{(i)}\left(y^{(i)}(w_0 + \mathbf{w}^T\mathbf{x}^{(i)}) - 1\right) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \mathbf{w}^T\sum_{i=1}^{N}\alpha^{(i)}y^{(i)}\mathbf{x}^{(i)} - w_0\sum_{i=1}^{N}\alpha^{(i)}y^{(i)} + \sum_{i=1}^{N}\alpha^{(i)} \qquad \alpha^{(i)} \geq 0$$

The original problem is now the same as minimization of L(**w**, w_0, α). We first minimize with respect to **w**, w_0, ($\alpha = [\alpha^{(1)}, \ldots, \alpha^{(N)}]$ is fixed). We take the gradient and find the optimal by setting the gradient to zero

$$\nabla_{\mathbf{w}} L(\mathbf{w}, w_0, \alpha) = \mathbf{w} - \sum_{i=1}^{N}\alpha^{(i)}y^{(i)}\mathbf{x}^{(i)} = 0 \qquad \mathbf{w} = \sum_{i=1}^{N}\alpha^{(i)}y^{(i)}\mathbf{x}^{(i)} \qquad \nabla_{w_0} L(\mathbf{w}, w_0, \alpha) = -\sum_{i=1}^{N}\alpha^{(i)}y^{(i)} = 0$$

Plugging these values back into L(w, w_0, α) we get the duel formalization

$$L(\alpha) = \frac{1}{2}\underbrace{\sum_{j=1}^{N}\alpha^{(j)}y^{(j)}\mathbf{x}^{(j)T}}_{\mathbf{w}^T}\underbrace{\sum_{i=1}^{N}\alpha^{(i)}y^{(i)}\mathbf{x}^{(i)}}_{\mathbf{w}} - \underbrace{\sum_{j=1}^{N}\alpha^{(j)}y^{(j)}\mathbf{x}^{(j)T}}_{\mathbf{w}^T}\underbrace{\sum_{i=1}^{N}\alpha^{(i)}y^{(i)}\mathbf{x}^{(i)}}_{\mathbf{w}} + \sum_{i=1}^{N}\alpha^{(i)} = -\frac{1}{2}\sum_{i,j=1}^{N}\alpha^{(j)}\alpha^{(i)}y^{(j)}y^{(i)}\mathbf{x}^{(j)T}\mathbf{x}^{(i)} + \sum_{i=1}^{N}\alpha^{(i)}$$

subject to $\sum_{i=1}^{N}\alpha^{(i)}y^{(i)} = 0$ and $\alpha^{(i)} \geq 0$

convex optimization problem

https://see.stanford.edu/materials/aimlcs229/cs229-notes3.pdf and
https://people.eecs.berkeley.edu/~jordan/courses/281B-spring04/lectures/lec2.pdf

SVM's can appear magical because we can transform the feature space $\Phi(\mathbf{x}^{(i)})$ (e.g. polynomial, RBF) and never have to compute $\Phi(\mathbf{x}^{(i)})$, instead we only need to compute the kernel $K(\mathbf{x}^{(i)},\mathbf{x}^{(j)}) = \Phi(\mathbf{x}^{(i)})^T \Phi(\mathbf{x}^{(j)})$ (In the description below we will not transform the feature space to keep the explanation clearer - but you can replace each $\mathbf{x}^{(i)}$ by $\Phi(\mathbf{x}^{(i)})$ yourself)

To show how this works, we transform our original problem into its duel formalization.

Given our constrained quadratic optimization problem: $\min \frac{1}{2}\|\mathbf{w}\|_2^2$ subject to $y^{(i)}(w_0 + \mathbf{w}^T\mathbf{x}^{(i)}) \geq 1$ for all i=1,...,N

We create the Lagrangian (each constraint is multiplied by a Lagrangian multiplier $\alpha^{(i)}$, for i = 1,...,N)

$$L(\mathbf{w}, w_0, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N}\alpha^{(i)}\left(y^{(i)}(w_0 + \mathbf{w}^T\mathbf{x}^{(i)}) - 1\right) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \mathbf{w}^T\sum_{i=1}^{N}\alpha^{(i)}y^{(i)}\mathbf{x}^{(i)} - w_0\sum_{i=1}^{N}\alpha^{(i)}y^{(i)} + \sum_{i=1}^{N}\alpha^{(i)} \qquad \alpha^{(i)} \geq 0$$

The original problem is now the same as minimization of L(**w**, w_0, α). We first minimize with respect to **w**, w_0, ($\alpha = [\alpha^{(1)}, \ldots, \alpha^{(N)}]$ is fixed). We take the gradient and find the optimal by setting the gradient to zero

$$\nabla_{\mathbf{w}} L(\mathbf{w}, w_0, \alpha) = \mathbf{w} = \sum_{i=1}^{N}\alpha^{(i)}y^{(i)}\mathbf{x}^{(i)} \qquad \nabla_{w_0} L(\mathbf{w}, w_0, \alpha) = -\sum_{i=1}^{N}\alpha^{(i)}y^{(i)} = 0$$

w is a linear sum of the training examples

Plugging these values back into L(**w**, w_0, α) we get the duel formalization

$$L(\alpha) = \frac{1}{2}\underbrace{\sum_{j=1}^{N}\alpha^{(j)}y^{(j)}\mathbf{x}^{(j)T}}_{\mathbf{w}^T}\underbrace{\sum_{i=1}^{N}\alpha^{(i)}y^{(i)}\mathbf{x}^{(i)}}_{\mathbf{w}} - \underbrace{\sum_{j=1}^{N}\alpha^{(j)}y^{(j)}\mathbf{x}^{(j)T}}_{\mathbf{w}^T}\underbrace{\sum_{i=1}^{N}\alpha^{(i)}y^{(i)}\mathbf{x}^{(i)}}_{\mathbf{w}} + \sum_{i=1}^{N}\alpha^{(i)} = -\frac{1}{2}\sum_{i,j=1}^{N}\alpha^{(j)}\alpha^{(i)}y^{(j)}y^{(i)}\mathbf{x}^{(j)T}\mathbf{x}^{(i)} + \sum_{i=1}^{N}\alpha^{(i)}$$

subject to $\sum_{i=1}^{N}\alpha^{(i)}y^{(i)} = 0$ and $\alpha^{(i)} \geq 0$

Quadratic form wrt α

The goal is to maximize

convex optimization problem

https://see.stanford.edu/materials/aimlcs229/cs229-notes3.pdf and
https://people.eecs.berkeley.edu/~jordan/courses/281B-spring04/lectures/lec2.pdf

Now we can send this to a quadratic programming solver!

There are many specialized QP solvers designed specifically for SVM's. We will not cover them :)

Equality constraints can be made from two inequality constraints

Many commercially available software packages for solving quadratic programming problems

$$\text{minimize}_{u \in R^L} : \frac{1}{2}\mathbf{u}^T Q \mathbf{u} + \mathbf{p}^T \mathbf{u}$$

$$\text{subject to} : \mathbf{a}^{(i)T}\mathbf{u} \geq c^{(i)} \quad (i = 1, \ldots, N)$$

Where the objective function is convex if Q is positive semidefinite: $\mathbf{u}^T Q \mathbf{u} \geq 0$ for all $\mathbf{u}$.

Can we turn our problem into this form?

Maximize:

$$L(\alpha) = -\frac{1}{2}\sum_{i,j=1}^{N}\alpha^{(j)}\alpha^{(i)}y^{(j)}y^{(i)}\mathbf{x}^{(j)T}\mathbf{x}^{(i)} + \sum_{i=1}^{N}\alpha^{(i)} \quad \text{subject to} \sum_{i=1}^{N}\alpha^{(i)}y^{(i)} = 0 \text{ and } \alpha^{(i)} \geq 0$$

Quadratic programming solvers usually are for minimization

$$\min_{\alpha} \frac{1}{2}\sum_{i,j=1}^{N}\alpha^{(j)}\alpha^{(i)}y^{(j)}y^{(i)}\mathbf{x}^{(j)T}\mathbf{x}^{(i)} - \sum_{i=1}^{N}\alpha^{(i)} \quad \text{subject to} \sum_{i=1}^{N}\alpha^{(i)}y^{(i)} = 0 \text{ and } \alpha^{(i)} \geq 0$$

$$\min_{\alpha} \frac{1}{2}\alpha^T \begin{bmatrix} y^{(1)}y^{(1)}\mathbf{x}^{(1)T}\mathbf{x}^{(1)} & y^{(1)}y^{(2)}\mathbf{x}^{(1)T}\mathbf{x}^{(2)} & \cdots & y^{(1)}y^{(N)}\mathbf{x}^{(1)T}\mathbf{x}^{(N)} \\ y^{(2)}y^{(1)}\mathbf{x}^{(2)T}\mathbf{x}^{(1)} & y^{(2)}y^{(2)}\mathbf{x}^{(2)T}\mathbf{x}^{(2)} & \cdots & y^{(2)}y^{(N)}\mathbf{x}^{(2)T}\mathbf{x}^{(N)} \\ \vdots & \vdots & & \vdots \\ y^{(N)}y^{(1)}\mathbf{x}^{(N)T}\mathbf{x}^{(1)} & y^{(N)}y^{(2)}\mathbf{x}^{(N)T}\mathbf{x}^{(2)} & \cdots & y^{(N)}y^{(N)}\mathbf{x}^{(N)T}\mathbf{x}^{(N)} \end{bmatrix} \alpha + \underbrace{(-1^T)\alpha}_{\text{linear term}}$$

quadratic term

subject to :

$$\underbrace{\mathbf{y}^T\boldsymbol{\alpha} = 0}_{\text{linear constraint}}$$

$$\underbrace{0 \leq \alpha^{(i)} \leq \infty}_{\text{range of values}}$$

$$\min_{\alpha} \frac{1}{2}\alpha^T Q \alpha - 1^T\alpha$$

subject to :

$$\mathbf{y}^T\alpha = 0$$

$$\alpha \geq 0$$

Instead of computing $\mathbf{x}^{(i)T}\mathbf{x}^{(j)}$ we could have computed $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ as long as K is symmetric and Q is pos. Semidefinite

We can perform the same calculations in Z-Space

Maximize:

$$L(\alpha) = -\frac{1}{2}\sum_{i,j=1}^{N}\alpha^{(j)}\alpha^{(i)}y^{(j)}y^{(i)}K(\mathbf{x}^{(j)},\mathbf{x}^{(i)}) + \sum_{i=1}^{N}\alpha^{(i)} \quad \text{subject to} \sum_{i=1}^{N}\alpha^{(i)}y^{(i)} = 0 \text{ and } \alpha^{(i)} \geq 0$$

Quadratic programming solvers usually are for minimization

$$\min_{\alpha} \frac{1}{2}\sum_{i,j=1}^{N}\alpha^{(j)}\alpha^{(i)}y^{(j)}y^{(i)}K(\mathbf{x}^{(j)},\mathbf{x}^{(i)}) - \sum_{i=1}^{N}\alpha^{(i)} \quad \text{subject to} \sum_{i=1}^{N}\alpha^{(i)}y^{(i)} = 0 \text{and } \alpha^{(i)} \geq 0$$

$$\min_{\alpha} \frac{1}{2}\alpha^{T}\begin{bmatrix} y^{(1)}y^{(1)}K(\mathbf{x}^{(1)},\mathbf{x}^{(1)}) & y^{(1)}y^{(2)}K(\mathbf{x}^{(1)},\mathbf{x}^{(2)}) & \cdots & y^{(1)}y^{(N)}K(\mathbf{x}^{(1)},\mathbf{x}^{(N)}) \\ y^{(2)}y^{(1)}K(\mathbf{x}^{(2)},\mathbf{x}^{(1)}) & y^{(2)}y^{(2)}K(\mathbf{x}^{(2)},\mathbf{x}^{(2)}) & \cdots & y^{(2)}y^{(N)}K(\mathbf{x}^{(2)},\mathbf{x}^{(N)}) \\ \vdots & \vdots & & \vdots \\ y^{(N)}y^{(1)}K(\mathbf{x}^{(N)},\mathbf{x}^{(1)}) & y^{(N)}y^{(2)}K(\mathbf{x}^{(N)},\mathbf{x}^{(2)}) & \cdots & y^{(N)}y^{(N)}K(\mathbf{x}^{(N)},\mathbf{x}^{(N)}) \end{bmatrix}\alpha + \underbrace{(-1^{T})\alpha}_{\text{linear term}}$$

quadratic term

subject to :

$\mathbf{y}^{T}\alpha = 0$ — linear constraint

$0 \leq \alpha^{(i)} \leq \infty$ — range of values

$$\min_{\alpha} \frac{1}{2}\alpha^{T}Q\alpha - 1^{T}\alpha$$

subject to :

$\mathbf{y}^{T}\alpha = 0$

$\alpha \geq 0$

For symmetric $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \phi(\mathbf{x}^{(i)})^{T}\phi(\mathbf{x}^{(j)})$ and positive semidefinite Q

"Kernel Trick"

Write the algorithm so the data only appeared when it was part of the dot product $\Phi\left(\mathbf{x}^{(i)}\right)^T \Phi\left(\mathbf{x}^{(j)}\right)$

➡ For some special types of transformations, we can efficiently compute $\Phi\left(\mathbf{x}^{(i)}\right)^T \Phi\left(\mathbf{x}^{(j)}\right)$ without transforming $\Phi\left(\mathbf{x}^{(i)}\right)$ or $\Phi\left(\mathbf{x}^{(j)}\right)$

- Remember the polynomial transformation?
- Define $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \Phi\left(\mathbf{x}^{(i)}\right)^T \Phi\left(\mathbf{x}^{(j)}\right)$
- We never need to create $\mathbf{z}^{(j)} = \Phi\left(\mathbf{x}^{(j)}\right)$

The kernel must satisfy Mercer's conditions (beyond the scope of this course)

This means…if I can compute, $\Phi\left(\mathbf{x}^{(i)}\right)^T \Phi\left(\mathbf{x}^{(j)}\right)$ efficiently I never need to worry about the length of $\Phi\left(\mathbf{x}^{(j)}\right)$

Learning in a transformed feature space:

Pair share: if we use $\Phi(\mathbf{x})$, do we need to actually perform the transformation?

Kernel trick: evaluate $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ without competing $\Phi(\mathbf{x}^{(j)})$ or $\Phi(\mathbf{x}^{(i)})$

Primal

$$\min_{\mathbf{w}, w_0} \|\mathbf{w}\|_2^2$$

subject to: $y^{(i)}(\mathbf{w}^T \Phi(\mathbf{x}^{(i)}) + w_0) \geq 1$ for all $i = 1 \ldots N$

Dual

$$\min_{\alpha} \sum_{i,j} \alpha^{(i)} \alpha^{(j)} y^{(i)} y^{(j)} K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) - \sum_{i} \alpha^{(i)}$$

$$\sum_{i=1}^{N} \alpha^{(i)} y^{(i)} = 0 \qquad \alpha^{(i)} \geq 0$$

We only need to know the result of $\Phi(\mathbf{x}^{(j)})^T \Phi(\mathbf{x}^{(i)})$

Prediction

$$g(\mathbf{x}) = \text{sign}\left(\mathbf{w}^T \Phi(\mathbf{x}) + w_0\right)$$

$$g(\mathbf{x}) = \text{sign}\left(\sum_{i \in I} \alpha^{(i)} y^{(i)} K(\mathbf{x}^{(i)}, \mathbf{x}) + w_0 \right)$$

We only need to know the result of $\Phi(\mathbf{x}^{(i)})^T \Phi(\mathbf{x})$

Support Vector $\mathbf{x}^{(i)}$

By computing the value of K without writing the transformation, we can have a computational advantage

The solution using the dual formalization

We turned our primal formula into the following minimization problem.

If we were using a kernel function this would become:

$$\min_{\alpha} \frac{1}{2}\sum_{i,j=1}^{N}\alpha^{(j)}\alpha^{(i)}y^{(j)}y^{(i)}K(\mathbf{x}^{(j)},\mathbf{x}^{(i)}) - \sum_{i=1}^{N}\alpha^{(i)}$$

subject to $\sum_{i=1}^{N}\alpha^{(i)}y^{(i)} = 0$ and $\alpha^{(i)} \geq 0$

What is returned by the solver is

$\alpha^T = (\alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(N)})$. We set $\mathbf{w} = \sum_{i=1}^{N}\alpha^{(i)}y^{(i)}\boldsymbol{x}^{(i)}$

and compute $w_0 = y^{(j)} - \sum_{i\in I}\alpha^{(i)}y^{(i)}K(\mathbf{x}^{(i)},\mathbf{x}^{(j)})$

$$g(\mathbf{x}) = \text{sign}\left(\sum_{i\in I}\alpha^{(i)}y^{(i)}K(\mathbf{x}^{(i)T},\mathbf{x}) + w_0\right)$$

The hypothesis for the set of training examples ((1,2.5), 1), ((0, 0.75), -1), ((1, 1), -1), ((2,2), 1), ((3,1), 1), ((2,3), 1), …

α^T = [0.9, 0, 1.4, 0, 0.5, 0, …, 0]

$$g(\mathbf{x}) = \text{sign}\left(\alpha^{(i)}y^{(i)}(1,1)\mathbf{x} + \alpha^{(j)}y^{(j)}(1,2.5)\mathbf{x} + \alpha^{(k)}y^{(k)}(3,1)\mathbf{x} + w_0\right)$$

$$g(\mathbf{x}) = \text{sign}\left(1.4(-1)(1,1)\mathbf{x} + 0.9(1)(1,2.5)\mathbf{x} + 0.5(1)(3,1)\mathbf{x} - 3.3\right)$$

Note that only the ***support vectors*** are in the hypothesis

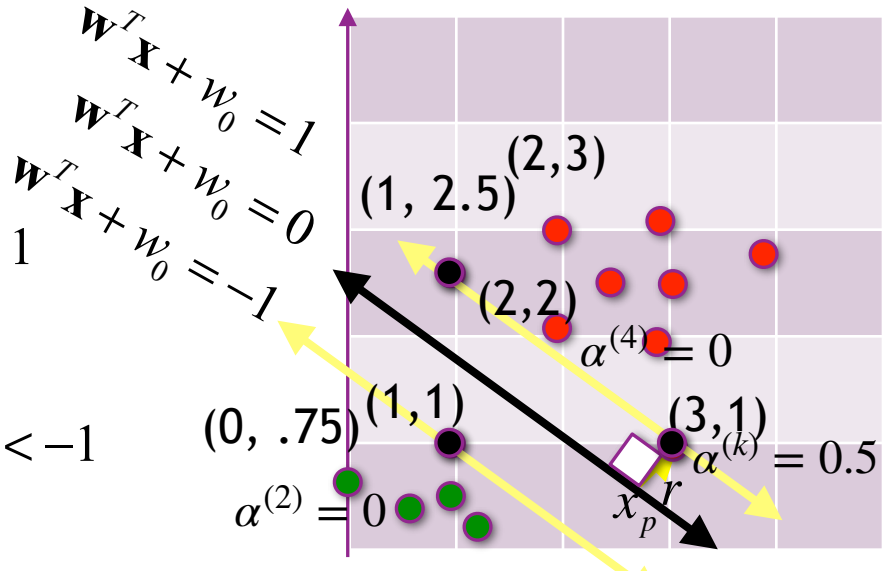
For any $\alpha^{(j)} \neq 0$ the $\mathbf{x}^{(j)}$ is on the margin and $y^{(j)}(w_0 + \mathbf{w}^T\mathbf{x}^{(j)}) = 1$

$$\mathbf{w}^T(1,2.5)^T + w_0 = 1$$

$$\mathbf{w}^T(2,2)^T + w_0 = 1$$

$$\mathbf{w}^T(0,0.75)^T + w_0 < -1$$

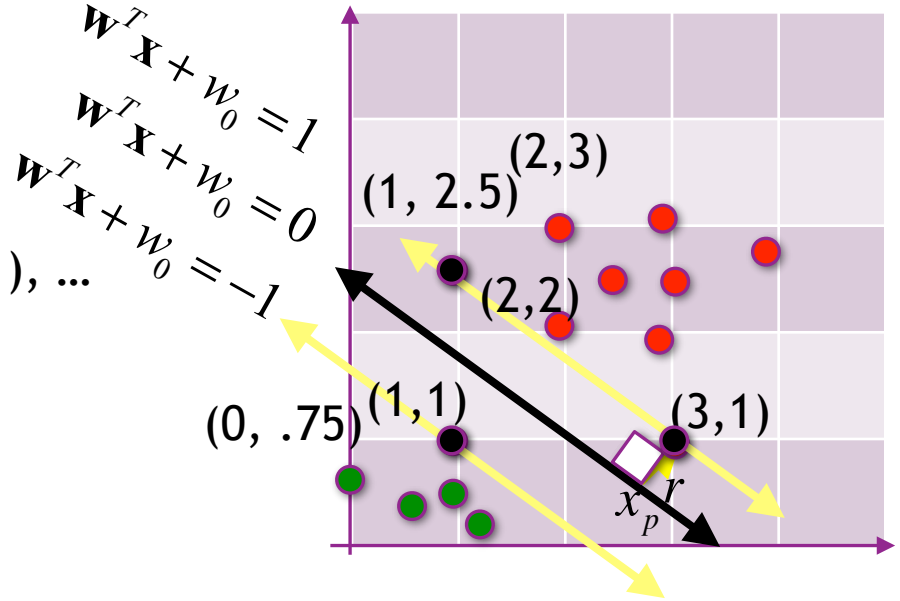
$$\mathbf{w}^T(2,3)^T + w_0 > 1$$



Support Vectors

The hypothesis for the set of training examples
((1,2.5), 1), ((0, 0.75), -1), ((1, 1), -1), ((2,2), 1), ((3,1), 1), ((2,3), 1), …

$$g(\mathbf{x}) = \text{sign}\left(\alpha^{(i)} y^{(i)} (1,1)\mathbf{x} + \alpha^{(j)} y^{(j)} (1,2.5)\mathbf{x} + \alpha^{(k)} y^{(k)} (3,1)\mathbf{x} + w_0\right)$$



Note that only the ***support vectors*** are in the hypothesis

$$\mathbf{w}^T(1,1)^T + w_0 = -1 \qquad \mathbf{w}^T(1,2)^T + w_0 = 1 \qquad \mathbf{w}^T(2,2)^T + w_0 = 1$$

$$\mathbf{w}^T(2,3)^T + w_0 > 1 \qquad \mathbf{w}^T(0,0.75)^T + w_0 < -1$$

Primal

In feature-space: $g(\mathbf{x}) = \text{sign}\left(\mathbf{w}^T\mathbf{x} + w_0\right)$

In Z-space (transformed feature space): $g(\mathbf{x}) = \text{sign}\left(\mathbf{w}^T\phi(\mathbf{x}) + w_0\right)$

Dual

$$g(\mathbf{x}) = \text{sign}\left(\sum_{i\in I} \alpha^{(i)} y^{(i)} (\mathbf{x}^{(i)T}\mathbf{x}) + w_0\right)$$

$$g(\mathbf{x}) = \text{sign}\left(\sum \alpha^{(i)} y^{(i)} K(\mathbf{x}^{(i)}, \mathbf{x}) + w_0\right)$$

Support Vector $\mathbf{x}^{(i)}$

Soft margin support vectors

MARGIN SUPPORT VECTORS

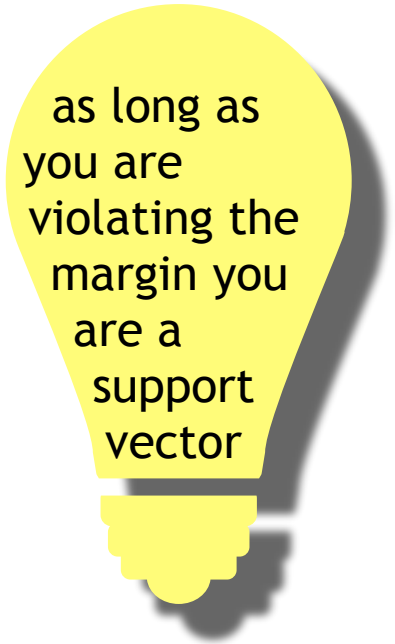
$$y^{(i)}(w_0 + \mathbf{w}^T \mathbf{x}^{(i)}) = 1$$

$$\xi^{(i)} = 0$$

$$\alpha^{(i)} > 0$$

NON-MARGIN SUPPORT VECTORS

$$y^{(i)}(w_0 + \mathbf{w}^T \mathbf{x}^{(i)}) < 1 \qquad \xi^{(i)} > 0$$



To decide on the value of C, use cross validation!

A good first guess is: $C = \frac{1}{N}$