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

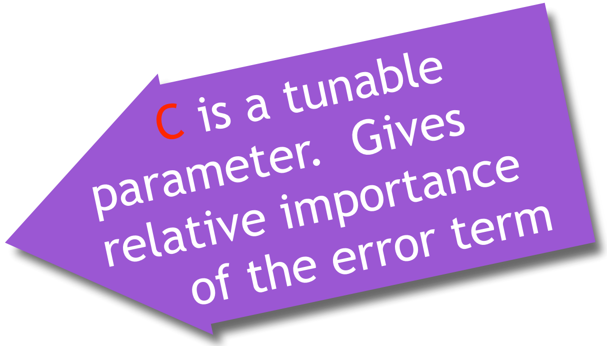
PROF. LINDA SELLIE

SOME SLIDES FROM PROF. RANGAN

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
  - ★ dual formula - a clever trick $g(\mathbf{x}) = \text{sign}\left(w_0 + \sum_{i \in I} \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)T} \mathbf{x}\right)$
  - ★ Dealing with non-linear data - feature transformation with the kernel trick - Show two popular feature maps

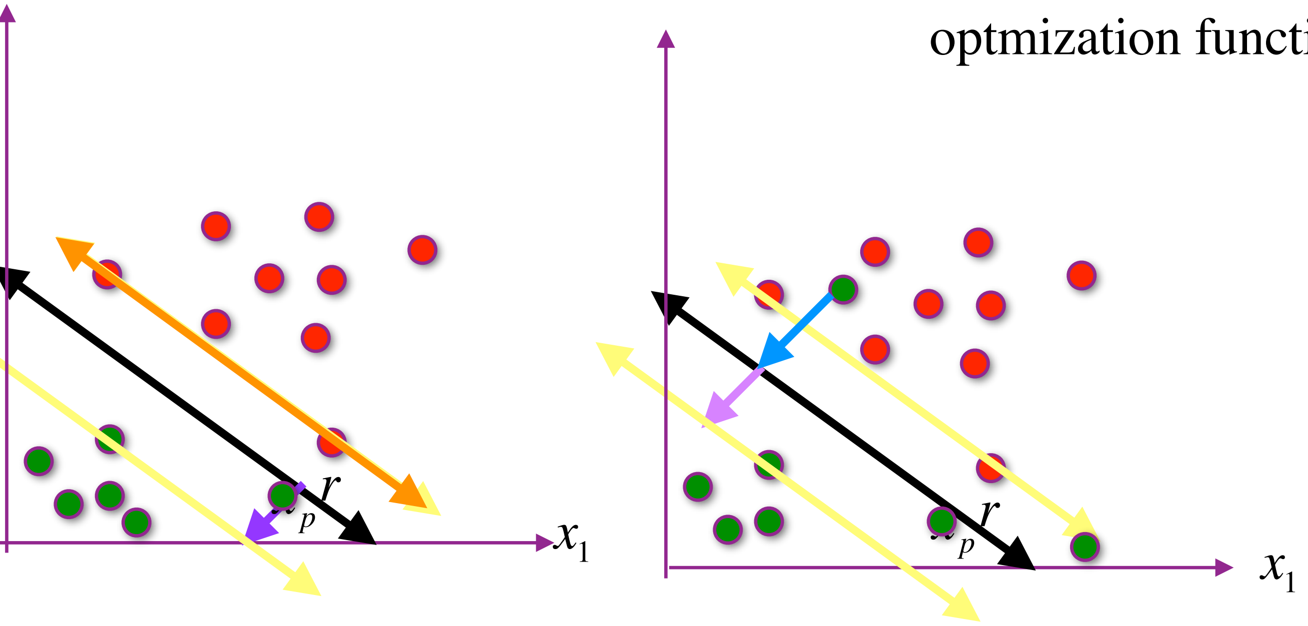
Soft-Margin SVM

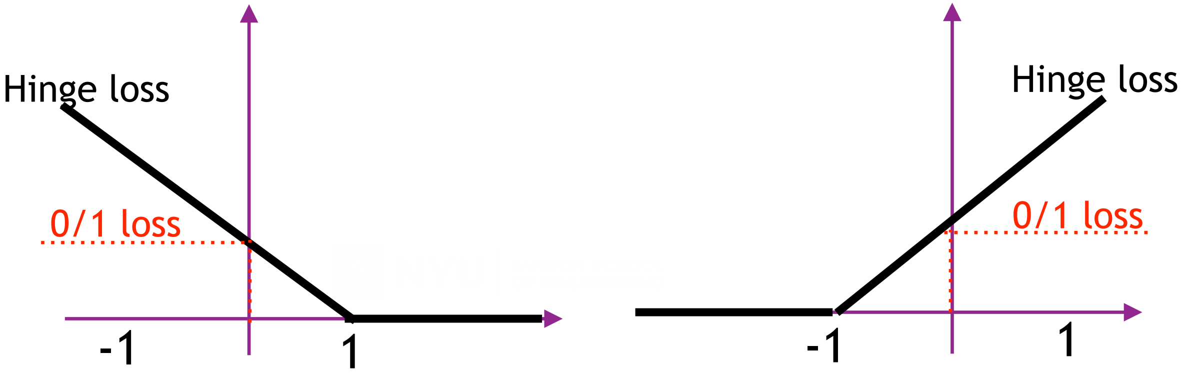


optmization function $\min_{w_0, \mathbf{w}, \{\xi^{(i)}\}_{i=1}^N} \|\mathbf{w}\|_2^2 + C\sum_{i=1}^{N} \xi^{(i)}$

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

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





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

optmization function

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

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

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

Pair share: What do you know about the functional margin for $\mathbf{x}$ if:

1) $\xi \geq 1$

2) $0 < \xi < 1$

3) $\xi = 0$

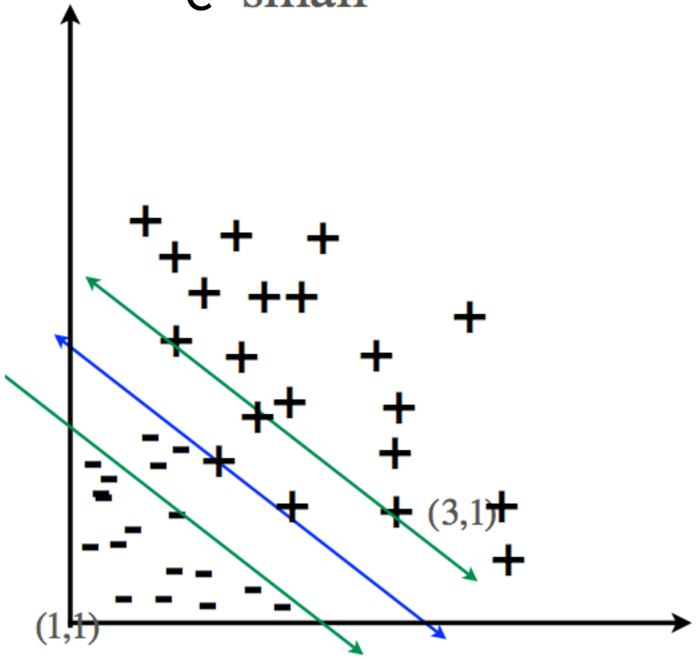
Pair share: Do you think that $\sum_{i=1}^{N} \xi^{(i)}$ is an upper bound on the number of training errors?

Pair share:

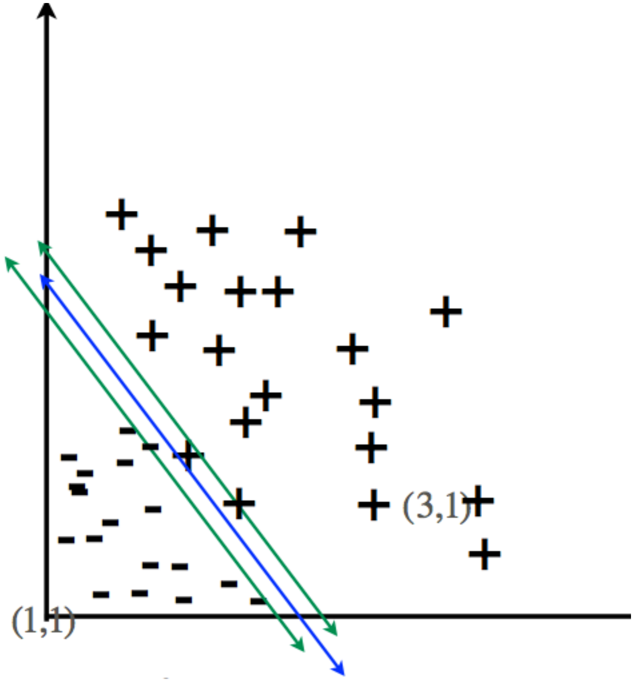
1) What happens to the margin if I make C large?

2) What happens to the margin if I make C small?

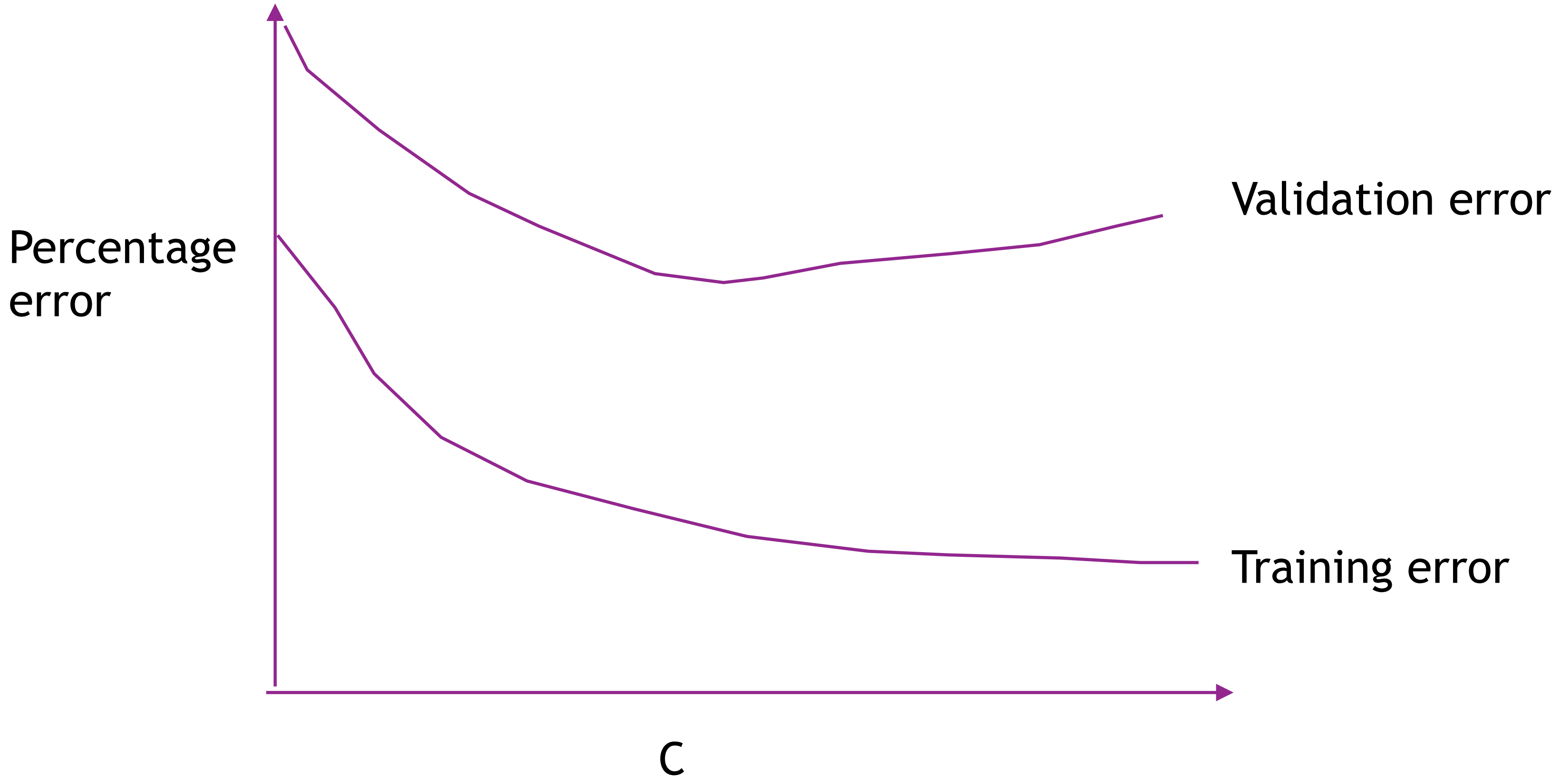
C small



Larger C

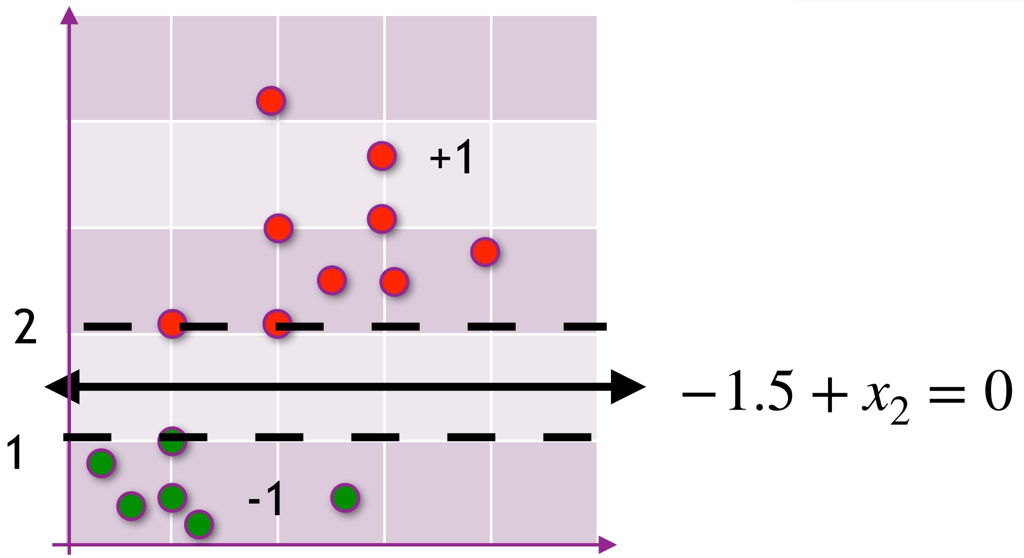


What if $C = \infty$?


Validation error
Percentage error
Training error
C

Example

Pair share: How can modify our decision boundary to have a functional margin of 1?



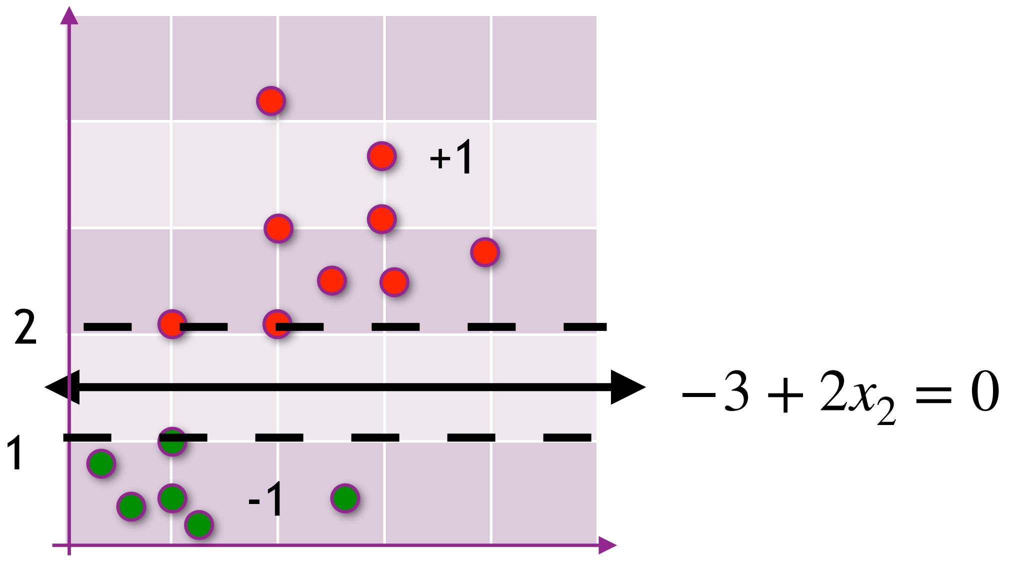
Decision boundary is $\mathbf{w} = [0,1]^T, w_0 = -1.5$

Is this the form we wanted?

The support vectors are supposed to have a functional margin of 1: $y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0) = 1$

Approach taken in this slide is from CMU 18-661

Example

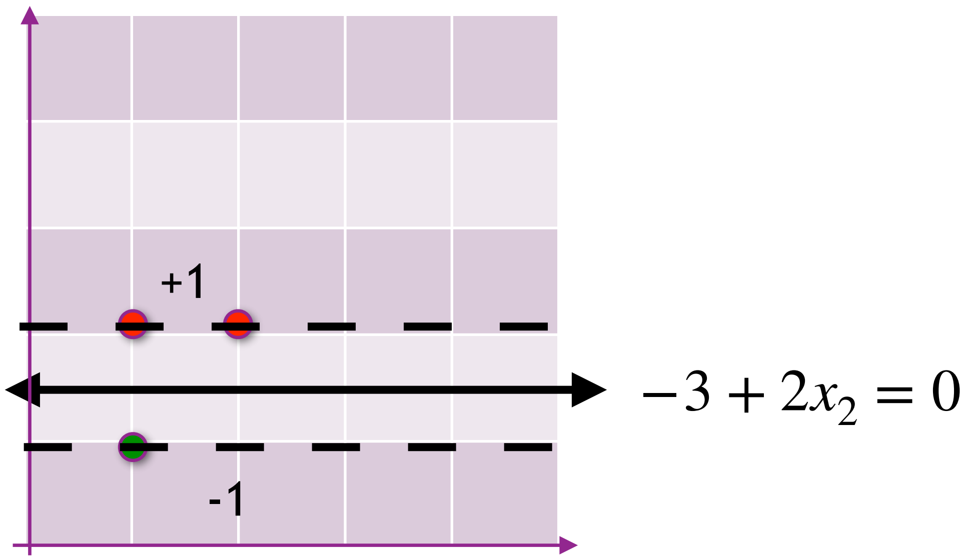


Decision boundary is $\mathbf{w} = [0,2]^T, w_0 = -3$

The support vectors have a functional margin of 1: $y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0) = 1$

Approach taken in this slide is from CMU 18-661

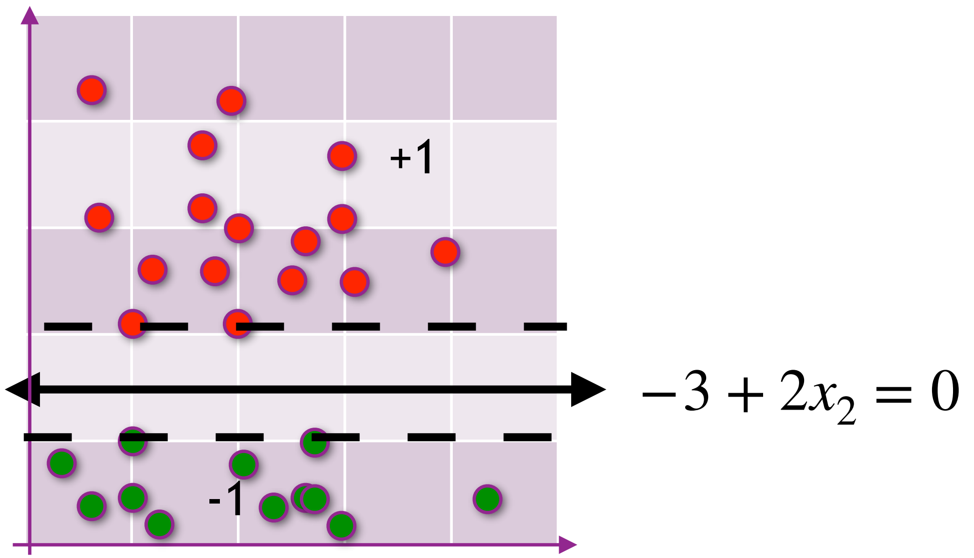
Example



The boundary doesn’t change if I remove points with a functional margin > 1

Approach taken in this slide is from CMU 18-661

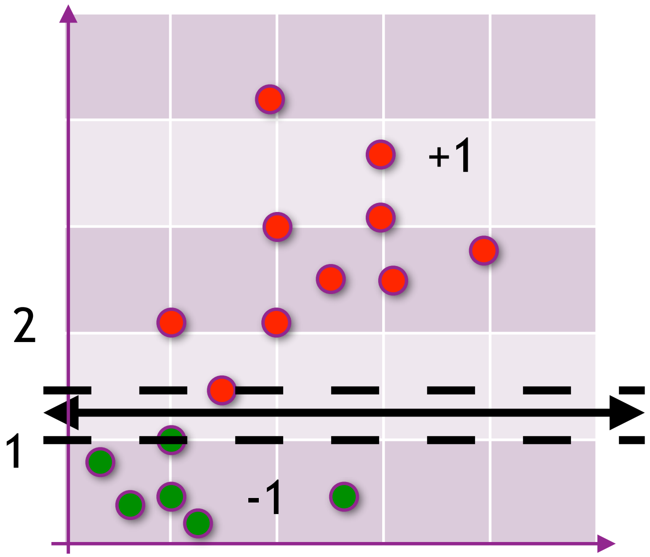
Example



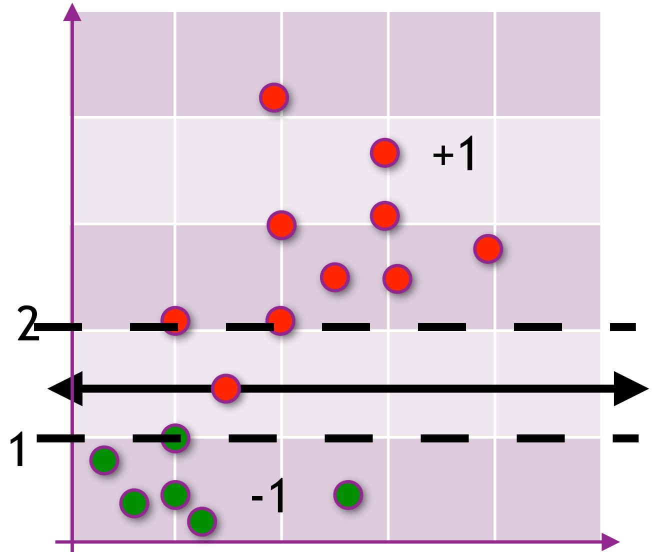
The solution doesn't change if I add points whose functional margin is ≥ 1

Approach taken in this slide is from CMU 18-661

Example



Our margin becomes smaller if we have an outlier



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

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

Approach taken in this slide is from CMU 18-661

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
  - ★ dual formula - a clever trick $g(\mathbf{x}) = \text{sign}\left(w_0 + \sum_{i \in I} \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)T} \mathbf{x}\right)$
  - ★ 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

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

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:

$$\min_{\mathbf{w}, w_0} \underbrace{\frac{1}{2} \|\mathbf{w}\|_2^2}_{\text{regularizer}} + C \sum_{i=1}^{N} \underbrace{\max\left(0, 1 - y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + w_0)\right)}_{\text{hinge Loss}}$$

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

Balance between loss function and regularizer.

We can use sub-gradient descent to find the optimal $\mathbf{w}$, w_0

The sub-gradient for the hinge loss will be 0, or $y^{(i)}\mathbf{x}^{(i)}$ depending on $y^{(i)}$ and $\mathbf{w}^T \mathbf{x}^{(i)} + w_0$

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

Rewriting our SVM objective function

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

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 $\lambda = 1/C$

$$\min_{\mathbf{w}, w_0} \frac{\lambda}{2} \underbrace{\|\mathbf{w}\|_2^2}_{\text{regularizer}} + \cancel{C} \sum_{i=1}^{N} \underbrace{\max\left(0, 1 - y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + w_0)\right)}_{\text{hinge Loss}}$$

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

Balance between loss function and regularizer.

We can use sub-gradient descent to find the optimal $\mathbf{w}$, w_0

The sub-gradient for the hinge loss will be 0, or $y^{(i)}\mathbf{x}^{(i)}$ depending on $y^{(i)}$ and $\mathbf{w}^T \mathbf{x}^{(i)} + w_0$

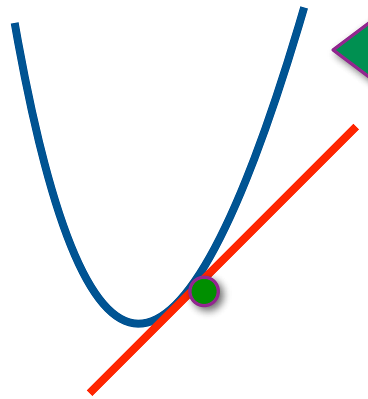
*In our optimizer, we will ignore the intercept term to make things easier

Our objective function is convex but not differentiable

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

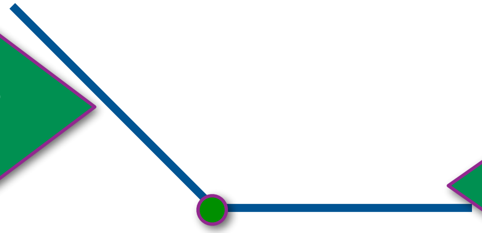
$\frac{\lambda}{2}\|\mathbf{w}\|_2^2$

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



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

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



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

?

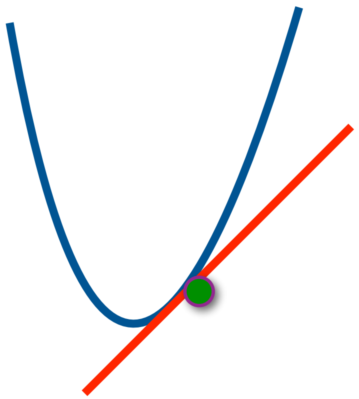
$$J(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \sum_{i=1}^{N} \max\left(0, 1 - y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + w_0)\right)$$

Derivative

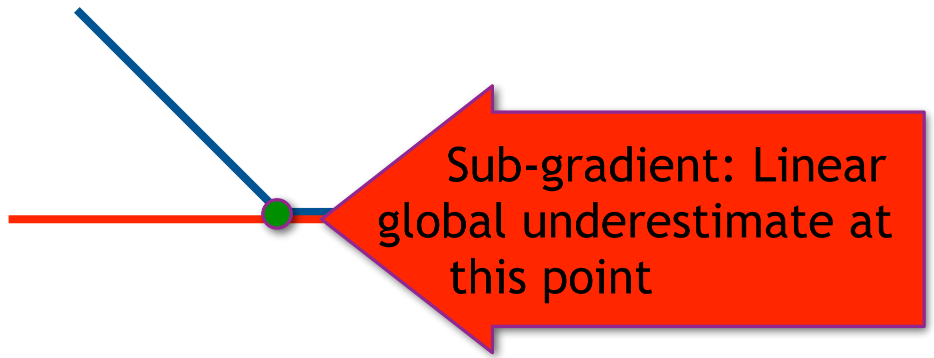
$\frac{\lambda}{2}\|\mathbf{w}\|_2^2$

Sub-derivative of the hinge loss

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



+



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

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

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

"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

If N is large, batch gradient is slow

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

The Pegasos Algorithm

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

$\mathbf{w}$ = random initialization

For t = 1,2,…,T:

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

Decrease the learning rate every iteration of the algorithm

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

Set

$$T = \tilde{O}\left(\frac{1}{\delta \lambda \epsilon}\right)$$

err w* $<$ w $+ \epsilon$
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.

The Pegasos Algorithm

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

$\mathbf{w}$ = 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

else

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

Set $T = \tilde{O}\left(\frac{1}{\delta\lambda\epsilon}\right)$

err w* $<$ w $+ \epsilon$ 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 objective function when the data is linearly separable
  - ★ Dealing with non-linear data - "Soft" margins for SVM - New objective function for the case where the data is not linearly separable
  - ★ dual formula - a clever trick

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

  - ★ Dealing with non-linear data - feature transformation with the kernel trick - Show two popular feature maps
- Multiclass

SVMs

- Maximizes distance of training data to boundary (built in regularization)
- Generalizes to nonlinear decision boundaries (I.e. feature transformation)
- Performs well with high dimensional data

Duel formula **motivation**: Polynomial Kernel

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

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

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

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

$$\Phi_2(\mathbf{x})^T\Phi_2(\mathbf{x}') = [1, x_1, x_2, x_1x_2, x_1^2, x_2^2] \cdot [1, x_1', x_2', x_1'x_2', x_1'^2, x_2'^2]$$

$$\Phi_2(\mathbf{x})^T\Phi_2(\mathbf{x}') = 1 + x_1x_1' + x_2x_2' + x_1x_2x_1'x_2' + x_1{}^2x_1'^2 + x_2{}^2x_2'^2$$

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

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

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

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

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

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

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

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

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

$$K((-1,1), (-0.75, -0.25)) = \underline{2.25}$$

Class exercise

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

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

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

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

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

Duel formula **motivation**: Polynomial Kernel

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

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

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

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

$\Phi_2(\mathbf{x})^T\Phi_2(\mathbf{x}') = [1, x_1, x_2, x_1x_2, x_1^2, x_2^2] \cdot [1, x_1', x_2', x_1'x_2', x_1'^2, x_2'^2]$

$\Phi_2(\mathbf{x})^T\Phi_2(\mathbf{x}') = 1 + x_1x_1' + x_2x_2' + x_1x_2x_1'x_2' + x_1{}^2x_1'^2 + x_2{}^2x_2'^2$

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

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

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

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

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

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

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

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

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 date features

$\Phi(-0.75, -0.25) = (1, \sqrt{}$

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

$K((-1,1), (-0.75, -0.25)) =$

2.25

class exercise

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

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

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

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

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

Primal, Dual SVM

- Learning a linear classifier $g(\mathbf{x}) = \text{sign}\left(\mathbf{w}^T \boxed{\mathbf{x}} + w_0\right)$ by solving an optimization problem over $\mathbf{w}, w_0$

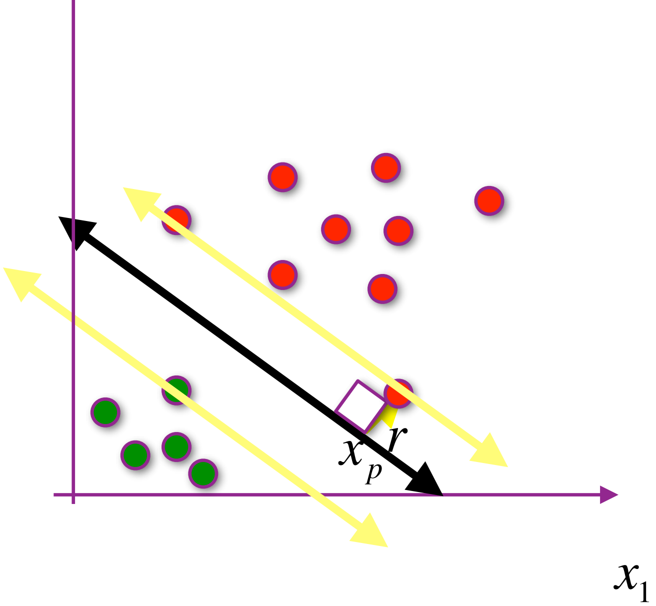
Known as the primal problem

- Instead, SMV can be formulated to learn the linear classifier $g(\mathbf{x}) = \text{sign}\left(\sum_{i \in I} \alpha^{(i)} y^{(i)} \boxed{\mathbf{x}^{(i)T}\mathbf{x}} + w_0\right)$ by solving an optimization problem over $\alpha^{(i)} \geq 0.$

Known as the dual problem

Where we write $\mathbf{w} = \sum_{i \in I} \alpha^{(i)} y^{(i)} \boxed{\mathbf{x}}$

Support Vectors



- In the hard margin case, the training examples that are on the margin **define the hyperplane**
- We can define the hyperplane as

$$\mathbf{w} = \sum_{i \in I} \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)}$$ where $i \in I$ if $\mathbf{x}^{(i)}$ is on the margin*

- The **classifier is defined by a few training examples**
- A similar property hold for the soft margin, where $i \in I$ if $\mathbf{x}^{(i)}$ is on the margin* or inside the margin

*Training examples where $\alpha^{(i)} \neq 0$ are the support vectors

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

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

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

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

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

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

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

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

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

"Kernel Trick"

Since 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 $\Phi\left(\mathbf{x}^{(j)}\right)$

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

Using Lagrange duality with SVM's

ORIGINAL PROBLEM

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

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

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

LAGRANGE DUALITY

If there is a valid solution, this is equal to:

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

Example: Given training examples ($\mathbf{x}^T$,y): ((1, 2.5),1), ((2, 2),1), ((3.1),1),..., ((0, 0.75),-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: $y^{(1)}\left(w_0 + w_1x_1^{(1)} + w_2x_2^{(1)}\right) \geq 1, y^{(2)}\left(w_0 + w_1x_1^{(2)} + w_2x_2^{(2)}\right) \geq 1, \cdots, y^{(N)}\left(w_0 + w_1x_1^{(N)} + w_2x_2^{(N)}\right) \geq 1$

The constrained quadratic optimization function can be rewritten as:

$$\min_{w_0,w_1,w_2} \frac{1}{2}[w_1, w_2]\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} + \max_{\alpha^{(1)} \geq 0} \alpha^{(1)}\left(1 - \underbrace{(w_0 + 1w_1 + 2.5w_2)}_{y^{(1)}\left(w_0 + w_1x_1^{(1)} + w_2x_2^{(1)}\right)}\right) + \max_{\alpha^{(2)} \geq 0} \alpha^{(2)}\left(1 - \underbrace{(w_0 + 2w_1 + 2w_2)}_{y^{(2)}\left(w_0 + w_1x_1^{(2)} + w_2x_2^{(2)}\right)}\right) + \ldots + \max_{\alpha^{(N)} \geq 0} \alpha^{(N)}\left(1 - \underbrace{(-w_0 + -1w_1 + -1w_2)}_{y^{(N)}\left(w_0 + w_1x_1^{(N)} + w_2x_2^{(N)}\right)}\right)$$

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