NYU Center for Data Science

DS-GA 1003
Machine Learning

Week 6: Lecture 6
Kernel Methods – Relationships between Features

NYU Center for Data Science

How can we add features without the issues surrounding wide datasets?

DS-GA 1003 Machine Learning

Week 6: Lecture 6

Kernel Methods – Relationships between Features

Adapted from Rosenberg, Rudin, Sontag, Boyd

Announcements

- Please check Week 6 agenda on NYU Classes
  - Homework 4
  - Midterm
    - Tuesday March 10 5:20-7:00PM
    - Review Slides
    - Practice Exam
    - Practice Problems

While you cannot bring your notes to the exam, you can bring a cheat-sheet

Discuss in Office Hours on Thursday

Review

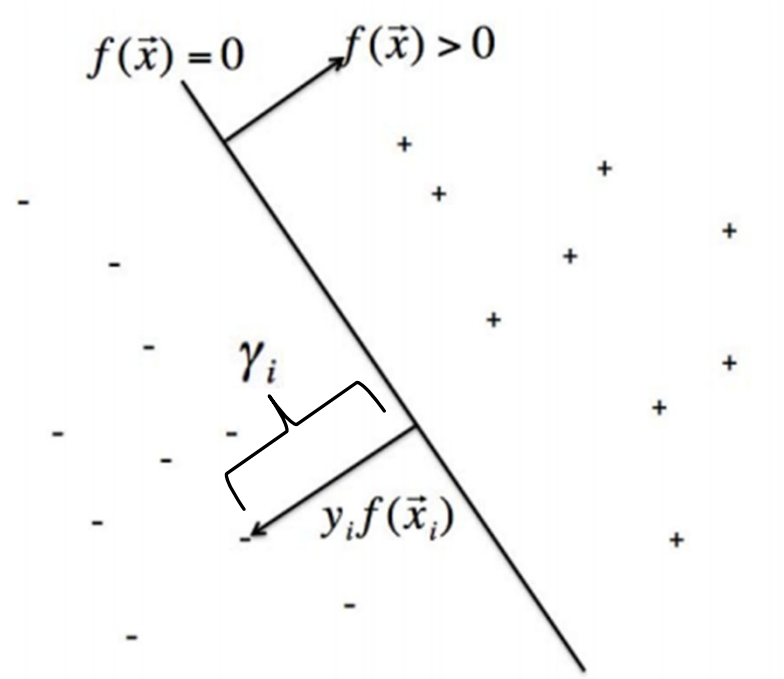
- Recall that margin intuitively means signed distance from decision boundary
- The decision boundary is the level set of f for value 0.
- For hypothesis f, we define the functional margin as y f(x)
- For hypothesis

$$f(\mathbf{x}) = \sum_{j=1}^{m} w^{(j)} x^{(j)} + w_0$$

we define the geometric margin as

$$\gamma_i = \frac{\mathbf{w}^T \mathbf{x}_i + w_0}{||\mathbf{w}||_2}$$

Sometimes we call f(x) the score. Both the sign and absolute value have information



Review

- We choose hypothesis so training points are far away from the decision boundary
- We want to maximize the minimum geometric margin

$$\gamma_i = \frac{\mathbf{w}^T \mathbf{x}_i + w_0}{||\mathbf{w}||_2}$$

by studying the problem

- We have a convex optimization problem with affine constraints. The Lagrangian is

$$L(\mathbf{w}, w_0, \alpha) = \frac{1}{2} \sum_{j=1}^{n} w^{(j)2} + \sum_{i=1}^{m} \alpha_i \left(1 - y_i \left(\mathbf{w}^T \mathbf{x}_i + w_0\right)\right)$$

Strict feasibility requires that the training data can be separated with a linear decision boundary

$$\min_{\mathbf{w}, w_0} \frac{1}{2} ||\mathbf{w}||_2^2 \text{ subject to } 0 \geq 1 - y_i \left(\mathbf{w}^T \mathbf{x}_i + w_0\right) \text{ for } i = 1, \ldots, m$$

Support Vector Machine

- Strict feasibility requires that the training data can be separated with a linear decision boundary
- First Order Optimality, Primal Feasibility, Dual Feasibility and Complementary Slackness give us conditions for solving the Primal / Dual Problem

$$L(\mathbf{w}, w_0, \alpha) = \frac{1}{2}\sum_{j=1}^{n} w^{(j)2} + \sum_{i=1}^{m} \alpha_i \left(1 - y_i\left(\mathbf{w}^T\mathbf{x}_i + w_0\right)\right)$$

Kuhn-Tucker Conditions

$$\nabla_{\mathbf{w}} L(\mathbf{w}, w_0, \alpha) = \mathbf{w} - \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial}{\partial \lambda_0} L(\mathbf{w}, w_0, \alpha) = -\sum_{i=1}^{m} \alpha_i y_i = 0 \Rightarrow \sum_{i=1}^{m} \alpha_i y_i = 0$$

$$\alpha_i \geq 0 \qquad -y_i\left(\mathbf{w}^T\mathbf{x}_i + w_0\right) + 1 \leq 0 \quad \text{for } i = 1, \ldots, m$$

$$\alpha_i\left(-y_i\left(\mathbf{w}^T\mathbf{x}_i + w_0\right) + 1\right) = 0 \text{ for } i = 1, \ldots, m$$

Review

- We can solve the quadratic programming problem with a package like CVXOPT.
- Another approach called Sequential Minimal Optimization applies coordinate descent to pairs of dual variables.

Dual of Hard Margin SVM

$$\max_{\boldsymbol{\alpha}} \mathcal{L}(\boldsymbol{\alpha})$$

where

$$\mathcal{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,k} \alpha_i \alpha_k y_i y_k \mathbf{x}_i^T \mathbf{x}_k$$

subject to constrains

$$\begin{cases} \alpha_i \geq 0 \quad i = 1 \ldots m \\ \sum_{i=1}^{m} \alpha_i y_i = 0 \end{cases}$$

Review

- For a support vector we have

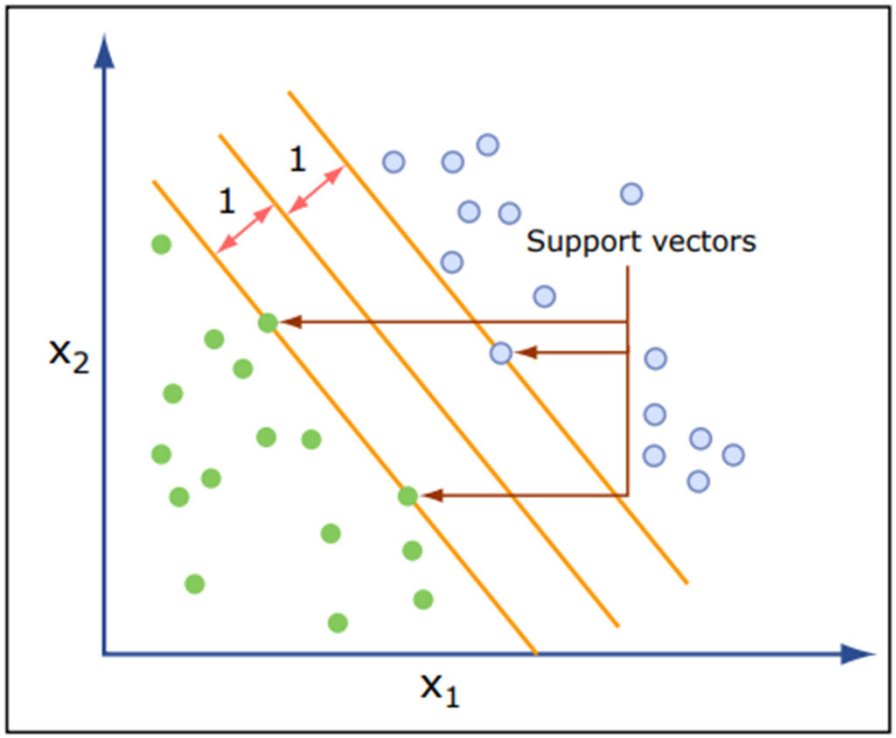
$$y_i \left(\mathbf{w}^{*T} \mathbf{x}_i + w_0^* \right) = 1$$

- For $y_i = 1$ we obtain

$$w_0^* = 1 - \mathbf{w}^{*T} \mathbf{x}_i$$

- We compute the dual variables α to get w before computing w_0

$$\begin{cases} \alpha_i^* > 0 & \Rightarrow \quad y_i f^*(\mathbf{x}_i) = \text{scaled margin}_i = 1 \\ 1 < y_i f^*(\mathbf{x}_i) & \Rightarrow \quad \alpha_i^* = 0 \end{cases}$$



Review

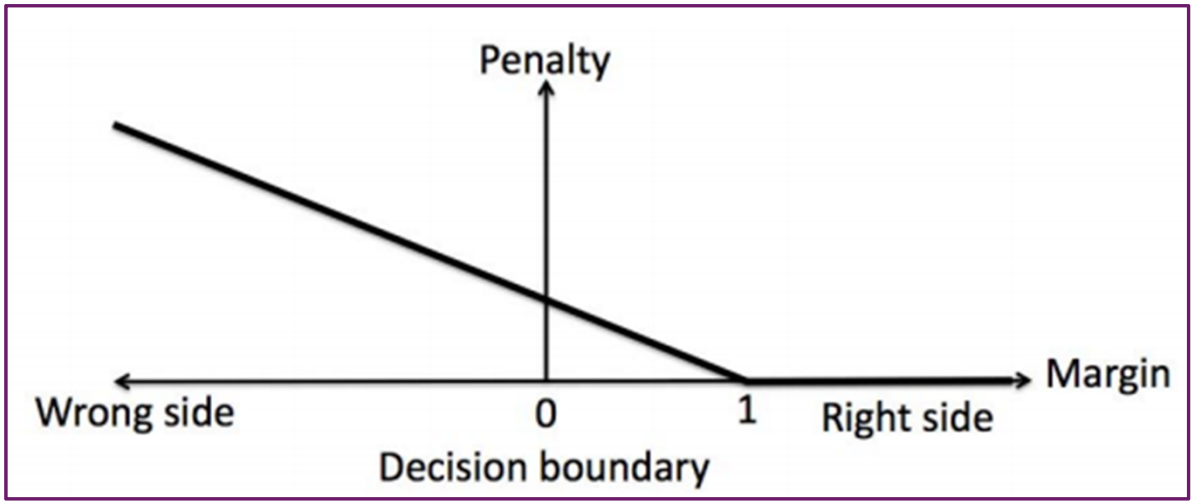
- We cannot separate some training sets with a linear decision boundary
- We can relax the constraint by adding a slack variable that captures the violation of the margin constraint.

$$\min_{\mathbf{w},w_0} \frac{1}{2}||\mathbf{w}||_2^2 + C\sum_{i=1}^{m} \xi_i$$

$$\text{subject to } \begin{cases} y_i\left(w^T\mathbf{x}_i + w_0\right) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases}$$

- Note that we can combine the two constraints to obtain

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



Review

Regularization

Empirical Risk

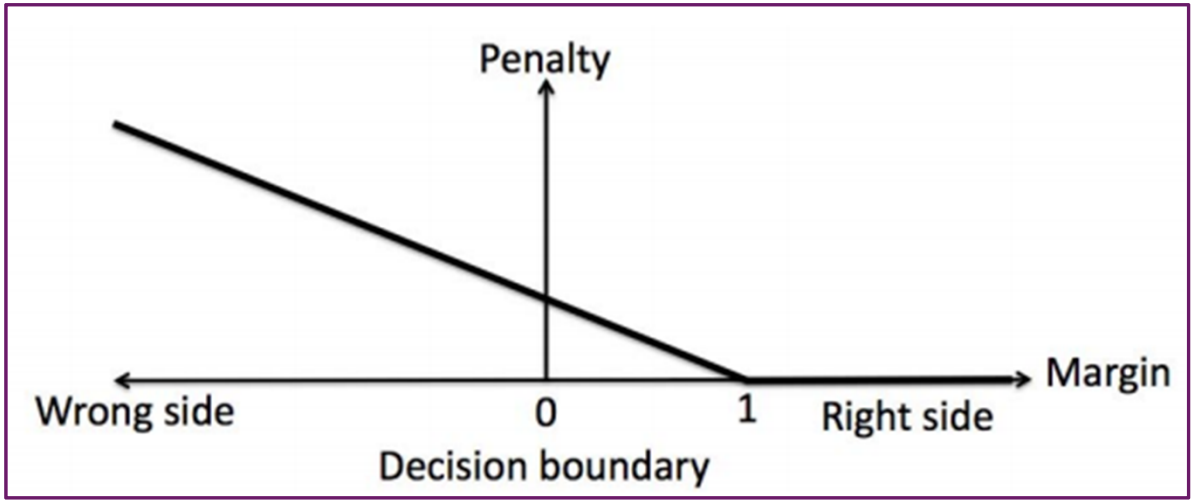
- We cannot separate some training sets with a linear decision boundary
- We can relax the constraint by adding a slack variable that captures the violation of the margin constraint.

$$\min_{\mathbf{w},w_0} \frac{1}{2}||\mathbf{w}||_2^2 + C\sum_{i=1}^{m} \xi_i$$

$$\text{subject to } \begin{cases} y_i\left(w^T\mathbf{x}_i + w_0\right) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases}$$

- Note that we can combine the two constraints to obtain

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



Review

- Unlike Hard Margin SVM, the Soft Margin SVM has an upper bound on the dual variables
- The dual variables tend to infinity for constraint violations in Hard Margin SVM
- The upper bound allows for misclassification in Soft Margin SVM by keeping the dual variables finite.

Dual of Soft Margin SVM

$$\max_{\boldsymbol{\alpha}} \mathcal{L}(\boldsymbol{\alpha})$$

where

$$\mathcal{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,k} \alpha_i \alpha_k y_i y_k \mathbf{x}_i^T \mathbf{x}_k$$

subject to constrains

$$\begin{cases} 0 \leq \alpha_i \leq C \quad i = 1 \dots m \\ \sum_{i=1}^{m} \alpha_i y_i = 0 \end{cases}$$

Agenda

- Support Vector Machines
  - Sparsity
  - Variants
- Kernels
  - Representer Theorem
- Training Support Vector Machines
  - Subgradients

References

- Shalev-Shwartz, *Understanding Machine Learning* (Chapter 16)
- Optional
  - D. Rosenberg, *Lecture Notes* (posted with slides)

Complementary Slackness

- We need two dual variables for Soft Margin Support Vector Machines
- Complementary slackness implied that

$$\alpha_i^* \left(1 - y_i f^*(x_i) - \xi_i^*\right) = 0$$

$$\lambda_i^* \xi_i^* = \left(\frac{c}{n} - \alpha_i^*\right) \xi_i^* = 0$$

Lagrange Multiplier	Constraint
λ_i	$-\xi_i \leqslant 0$
α_i	$(1 - y_i f(x_i)) - \xi_i \leqslant 0$

If $y_i f^*(x_i) > 1$ then the margin loss is $\xi_i^* = 0$, and we get $\alpha_i^* = 0$.

If $y_i f^*(x_i) < 1$ then the margin loss is $\xi_i^* > 0$, so $\alpha_i^* = \frac{c}{n}$.

If $\alpha_i^* = 0$, then $\xi_i^* = 0$, which implies no loss, so $y_i f^*(x) \geqslant 1$.

If $\alpha_i^* \in \left(0, \frac{c}{n}\right)$, then $\xi_i^* = 0$, which implies $1 - y_i f^*(x_i) = 0$.

Complementary Slackness

- We need two dual variables for Soft Margin Support Vector Machines
- Complementary slackness implied that

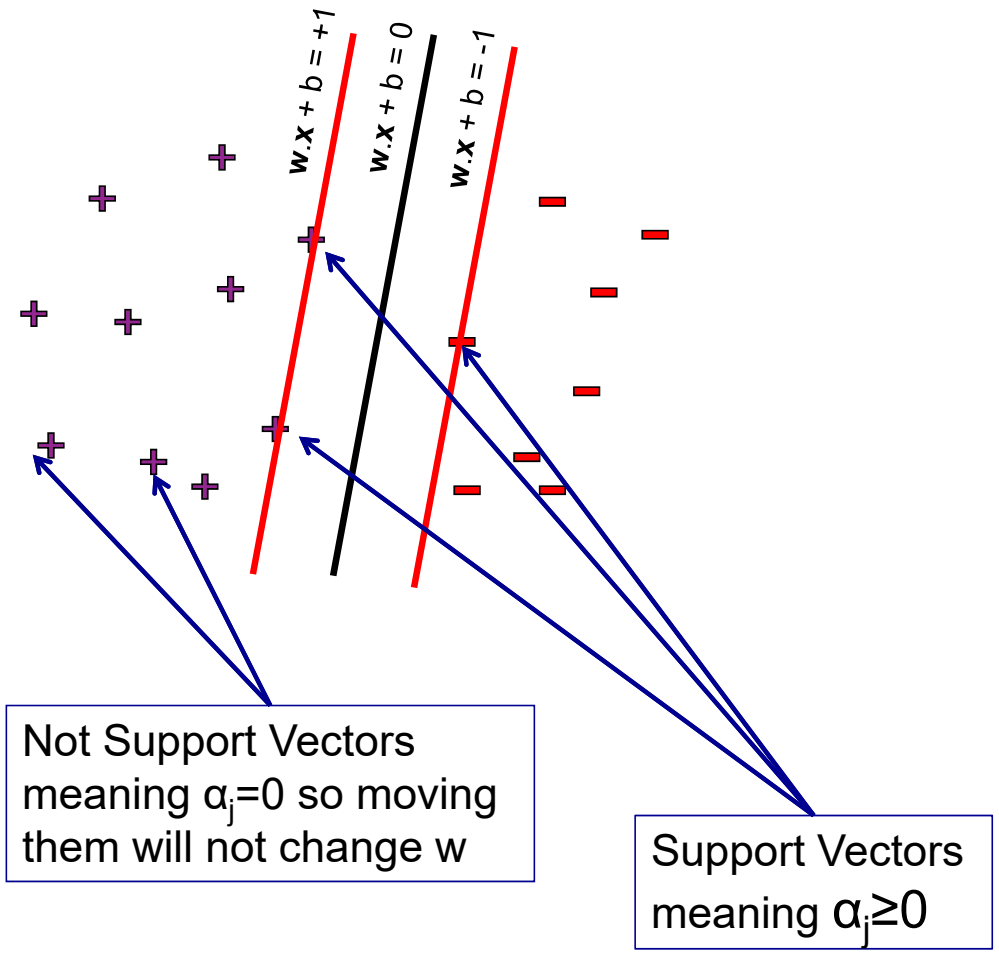
$$\alpha_i^* \left(1 - y_i f^*(x_i) - \xi_i^*\right) = 0$$

$$\lambda_i^* \xi_i^* = \left(\frac{c}{n} - \alpha_i^*\right) \xi_i^* = 0$$

$$\begin{aligned}
\alpha_i^* = 0 &\implies y_i f^*(x_i) \geq 1 \\
\alpha_i^* \in \left(0, \frac{c}{n}\right) &\implies y_i f^*(x_i) = 1 \\
\alpha_i^* = \frac{c}{n} &\implies y_i f^*(x_i) \leq 1
\end{aligned}$$

$$\begin{aligned}
y_i f^*(x_i) < 1 &\implies \alpha_i^* = \frac{c}{n} \\
y_i f^*(x_i) = 1 &\implies \alpha_i^* \in \left[0, \frac{c}{n}\right] \\
y_i f^*(x_i) > 1 &\implies \alpha_i^* = 0
\end{aligned}$$

Sparsity for Support Vector Machines



- Kuhn Tucker conditions showed that the weights are in the span of the data

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

- Most of the dual variables tend to be zero. Other dual variables correspond to training data that does not affect the weights
- You can omit the data without affecting predictions

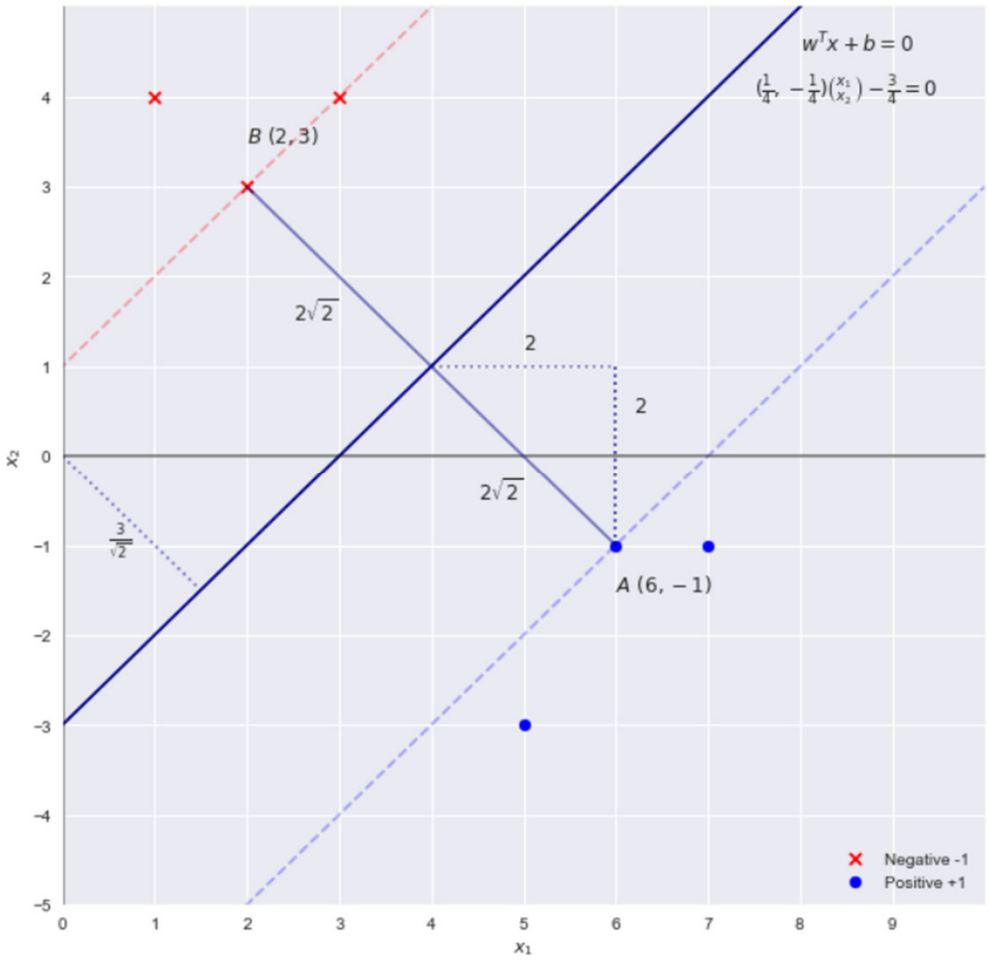
Calculating Support Vectors

```
import numpy as np
from sklearn.svm import SVC

X = np.array([[3,4],[1,4],[2,3],[6,-1],[7,-1],[5,-3]] )
y = np.array([-1,-1, -1, 1, 1 , 1 ])

clf = SVC(C = 1e5, kernel = 'linear')
clf.fit(X, y)
```

```
SVC(C=100000.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='linear', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
```

```
clf.support_vectors_
```

```
array([[ 2.,  3.],
       [ 6., -1.]])
```



Calculating Support Vectors

- Equation for the decision boundary is

$$w = [1, -1] \;\; b = -3$$

- Note that we can scale the weights but obtain the same decision boundary

$$cx_1 + cx_2 - 3c = 0$$

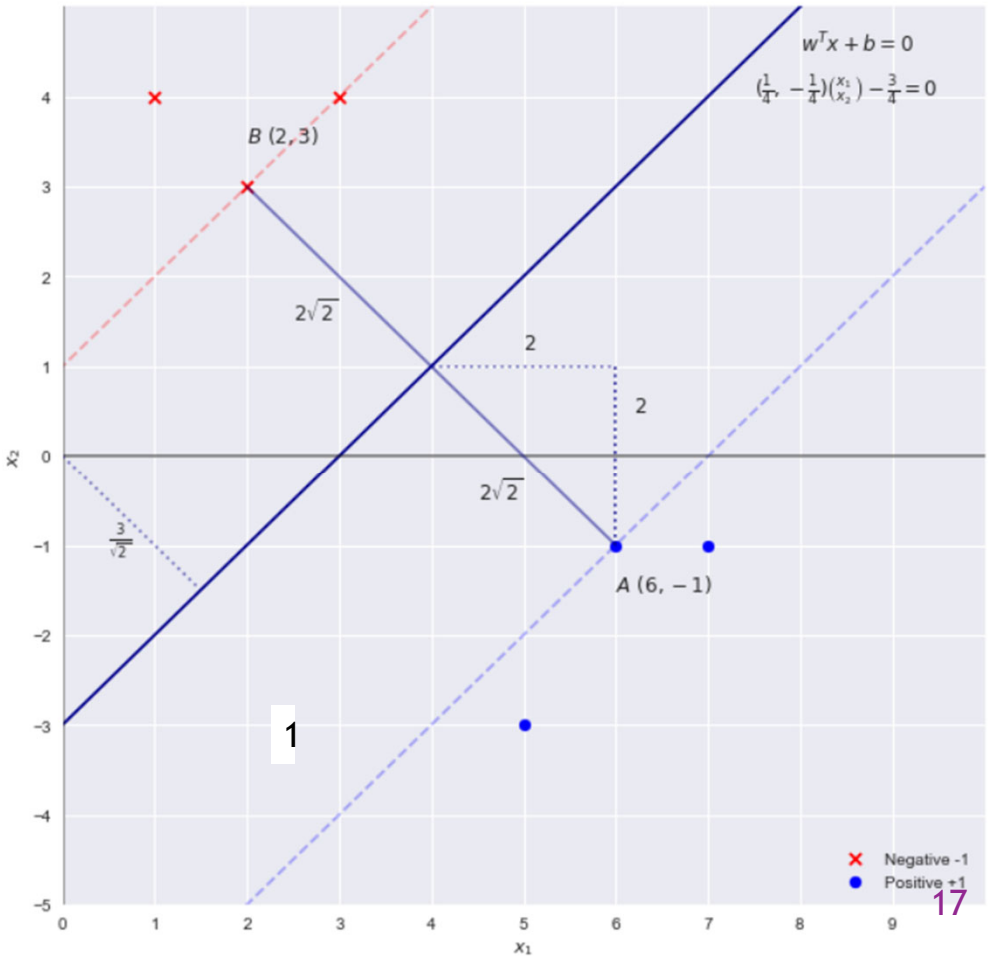
$$w = [c, -c], \; b = -3c$$

- Choosing c to have the indicated geometric margin

$$\frac{2}{||w||} = 4\sqrt{2}$$

$$\frac{2}{\sqrt{2}c} = 4\sqrt{2}$$

$$c = \frac{1}{4}$$



Calculating Support Vectors

- Using two of the Kuhn Tucker conditions

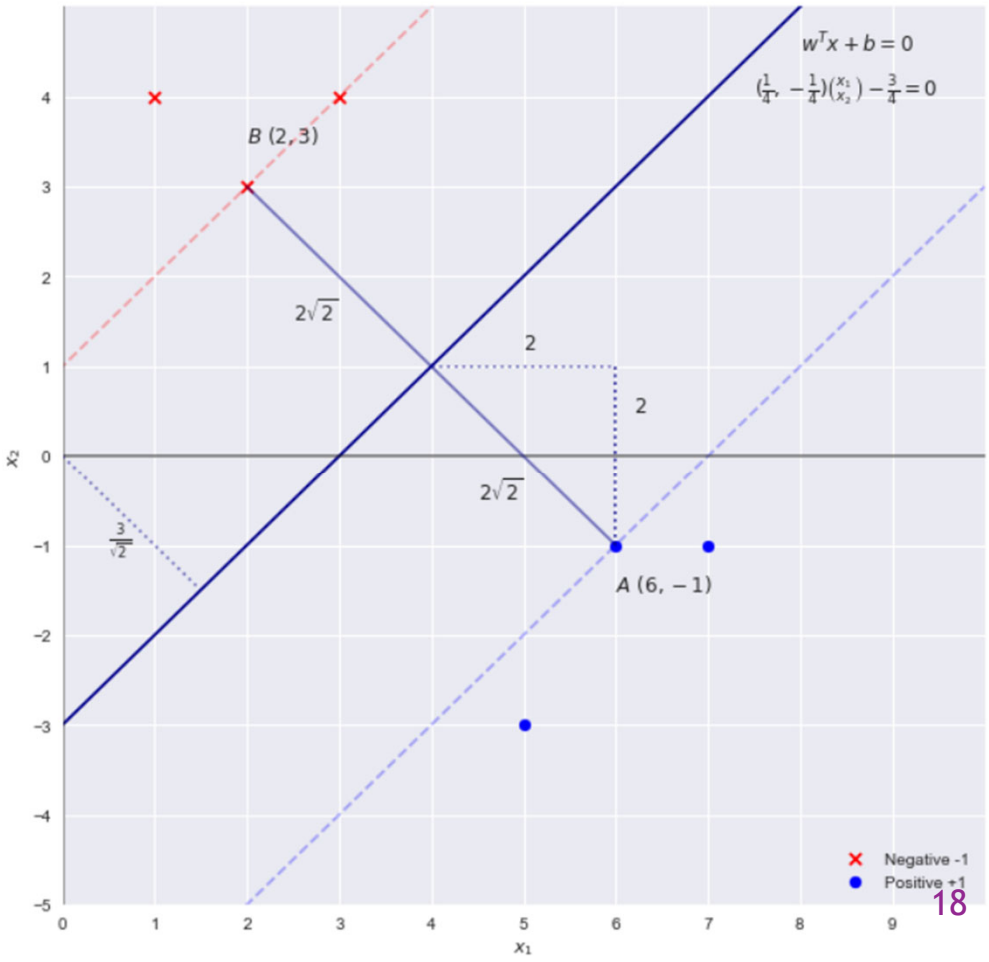
$$w = \sum_{i}^{m} \alpha_i y^{(i)} x^{(i)}$$

$$\sum_{i}^{m} \alpha_i y^{(i)} = 0$$

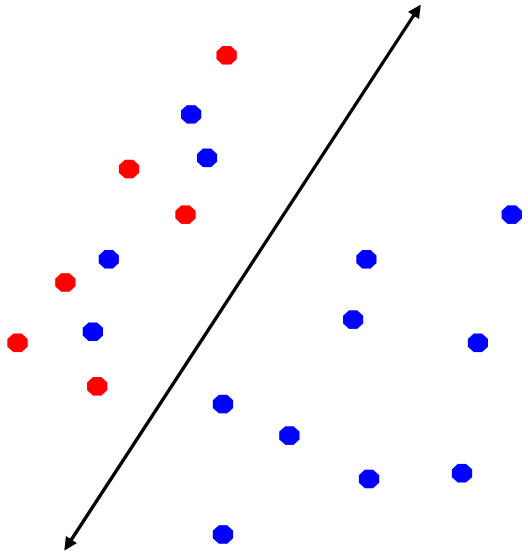
we can solve a system of equations

$$\begin{bmatrix} 6\alpha_1 - 2\alpha_2 - 3\alpha_3 \\ -1\alpha_1 - 3\alpha_2 - 4\alpha_3 \\ 1\alpha_1 - 1\alpha_2 - 1\alpha_3 \end{bmatrix} = \begin{bmatrix} 1/4 \\ -1/4 \\ 0 \end{bmatrix}$$

$$\alpha = \begin{bmatrix} 1/16 \\ 1/16 \\ 0 \end{bmatrix}$$



Imbalanced Data



$$N = N_+ + N_-$$

$$\min_{w,b} \; ||w||_2^2 + \frac{CN}{2N_+} \sum_{j:y_j=+1} \xi_j + \frac{CN}{2N_-} \sum_{j:y_j=-1} \xi_j$$

Class-specific weighting of the slack variables

- We encounter imbalanced data in practice. Here some classes have more points than other classes.
- If we want errors to be equally distributed between the classes, then we might need to change the dataset through over-sampling or under-sampling
- For SVM a slight modification allows us to avoid changing the dataset

Transfer Learning

- Suppose we want to determine spam classifier like Homework 1 for David
- David has few labelled emails. Chris has many labelled emails.
- Assuming that David and Chris have related notions of spam, how can we use Chris' data or classifier?

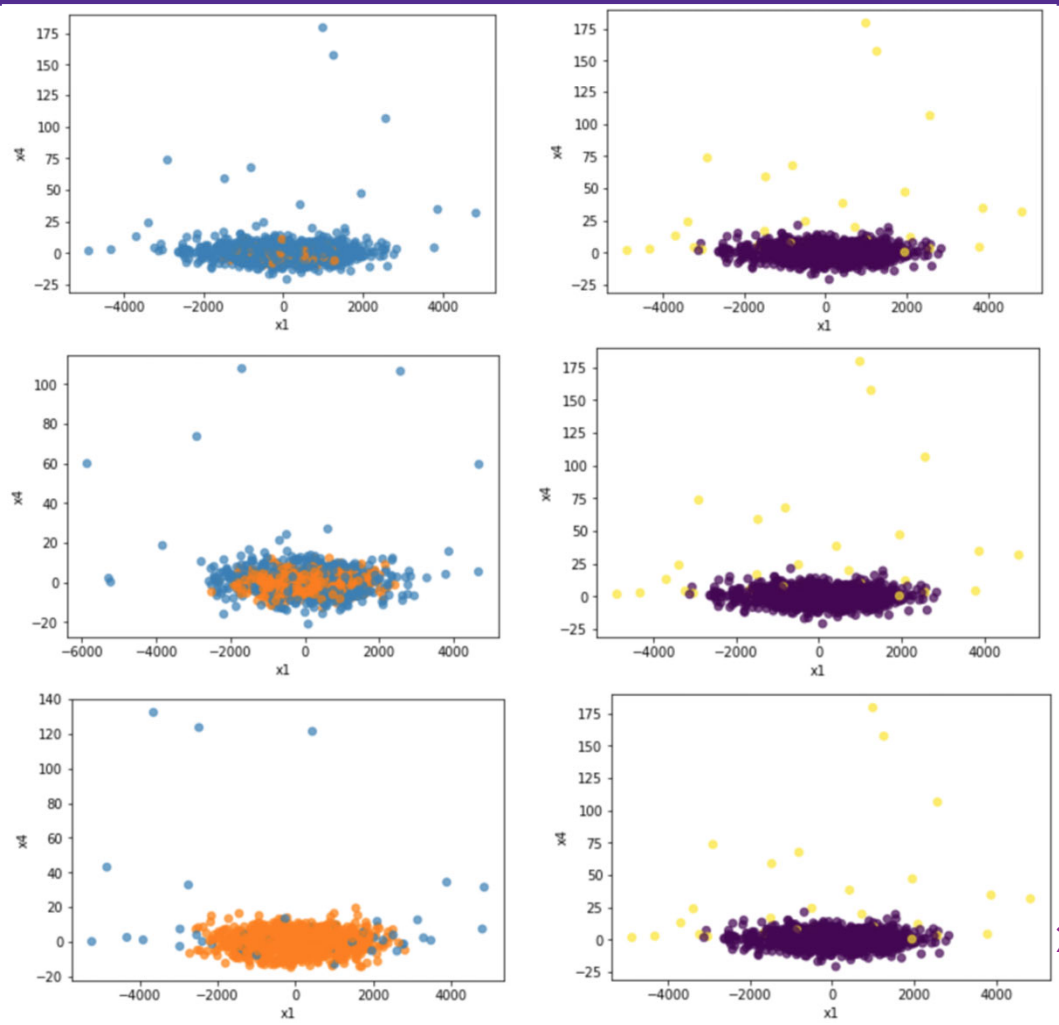
$$\min_{\mathbf{w}_d, b_d} \frac{C}{|D_d|} \sum_{\mathbf{x}, y \in D_d} \max(0, 1 - y(\mathbf{w}_d^T \mathbf{x} + b_d)) + \frac{1}{2}\|\mathbf{w}_d - \mathbf{w}_c\|^2$$

Could we try to…

- Average weights between David and Chris?
- Combine emails. Duplicating emails of David.
- Modify the SVM objective to use Chris' weights. This is transfer learning

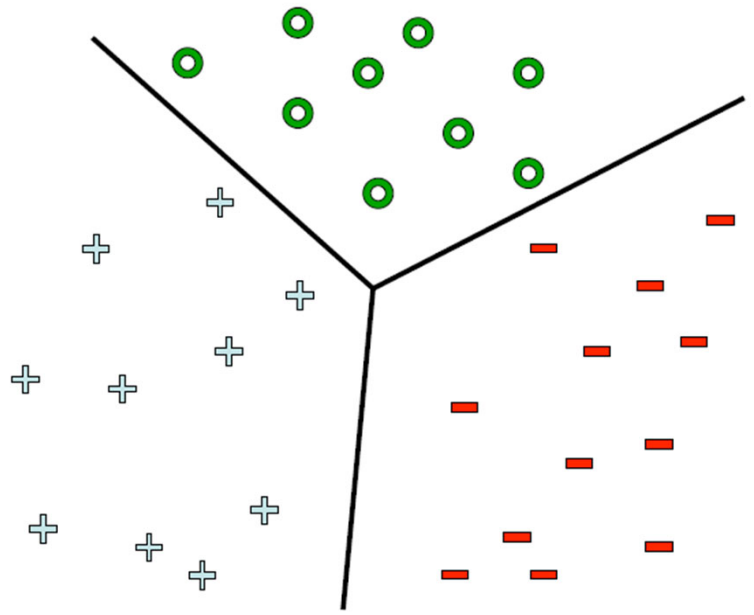
One Class Classification

- We can use a variant of the objective for Soft Margin Support Vector Machines to determine anomalies in datasets
- One Class Classification is an approach to unsupervised learning where we identify outliers through violations of the margin constraint



Multiclass Classification

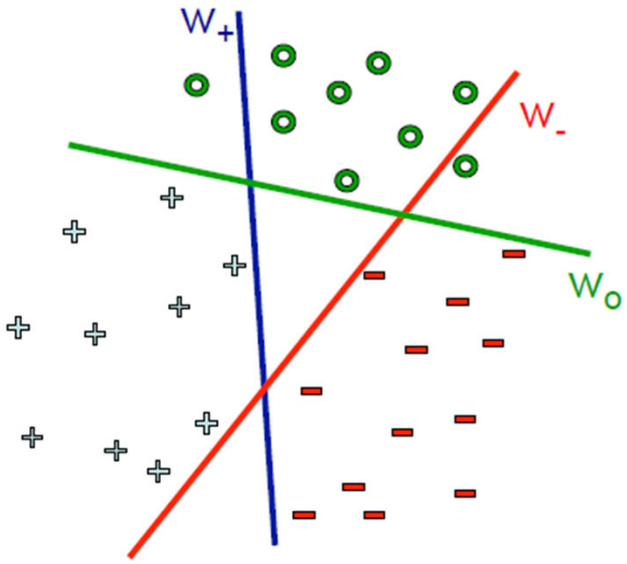
- Can we use SVM to classify into three or more classes? We need it for MNIST data.
- Before studying multiclass classification in detail, we can try two approaches that reduce many classes to two classes
  - One-vs-All
  - One-vs-One



Sometimes called One-vs-Rest and All-vs-All

Multiclass Classification

- Can we use SVM to classify into three or more classes? We need it for MNIST data.
- Before studying multiclass classification in detail, we can try two approaches that reduce many classes to two classes
  - One-vs-All
  - One-vs-One



Learn 3 classifiers:

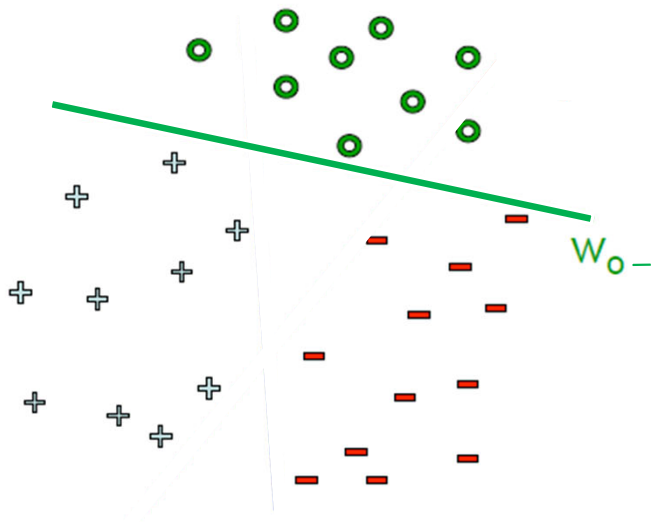
- \- vs {o,+}, weights w_-
- \+ vs {o,-}, weights w_+
- o vs {+,-}, weights w_o

Predict label using:

$$\hat{y} \leftarrow \arg\max_k \; w_k \cdot x + b_k$$

Multiclass Classification

- Can we use SVM to classify into three or more classes? We need it for MNIST data.
- Before studying multiclass classification in detail, we can try two approaches that reduce many classes to two classes
  - One-vs-All
  - One-vs-One



Learn 3 Classifiers:

- - vs + weights w_{+-}
- - vs o weights w_{o-}
- + vs o weights w_{o+}

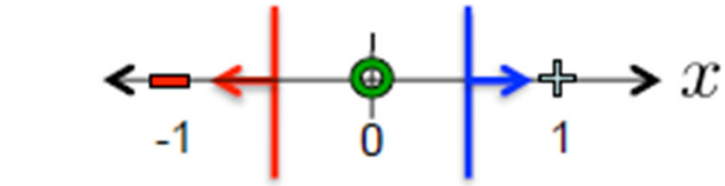
Predict Label using:

$\text{argmax}_i \ \sum_j w_{ij} x + b_{ij}$

Note that
$w_{ij} x + b_{ij} = -w_{ji} x - b_{ji}$

Multiclass Classification

- Can we use SVM to classify into three or more classes? We need it for MNIST data.
- Before studying multiclass classification in detail, we can try two approaches that reduce many classes to two classes
  - One-vs-All
  - One-vs-One



$w_- = -1$ $\quad$ $w_+ = 1$

$b_- = -.5$ $\quad$ $b_+ = -.5$

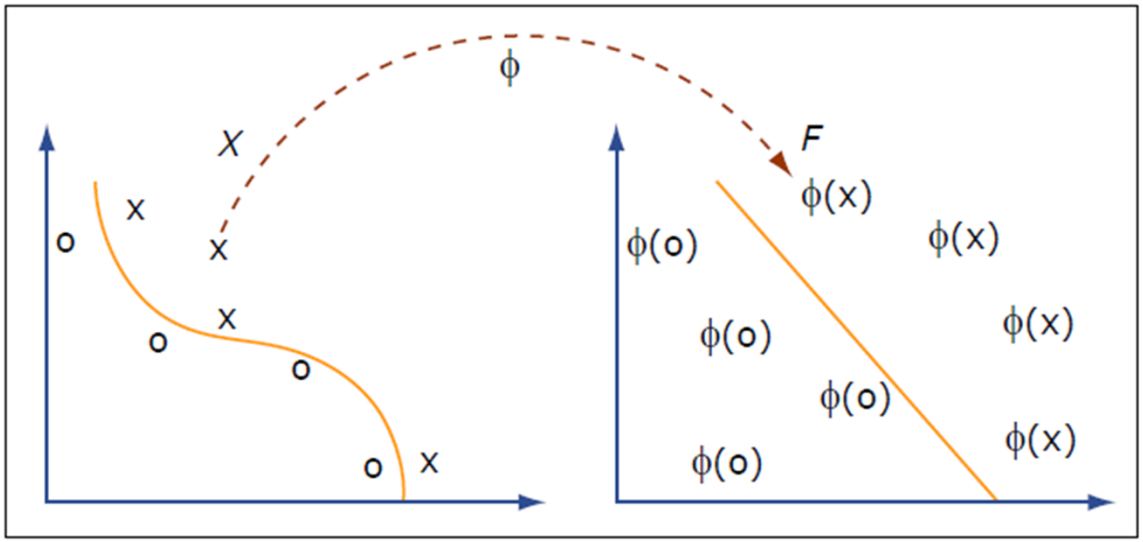
$w_o = 0$

$b_o = .001$

Could we separate the data with One-vs-All?

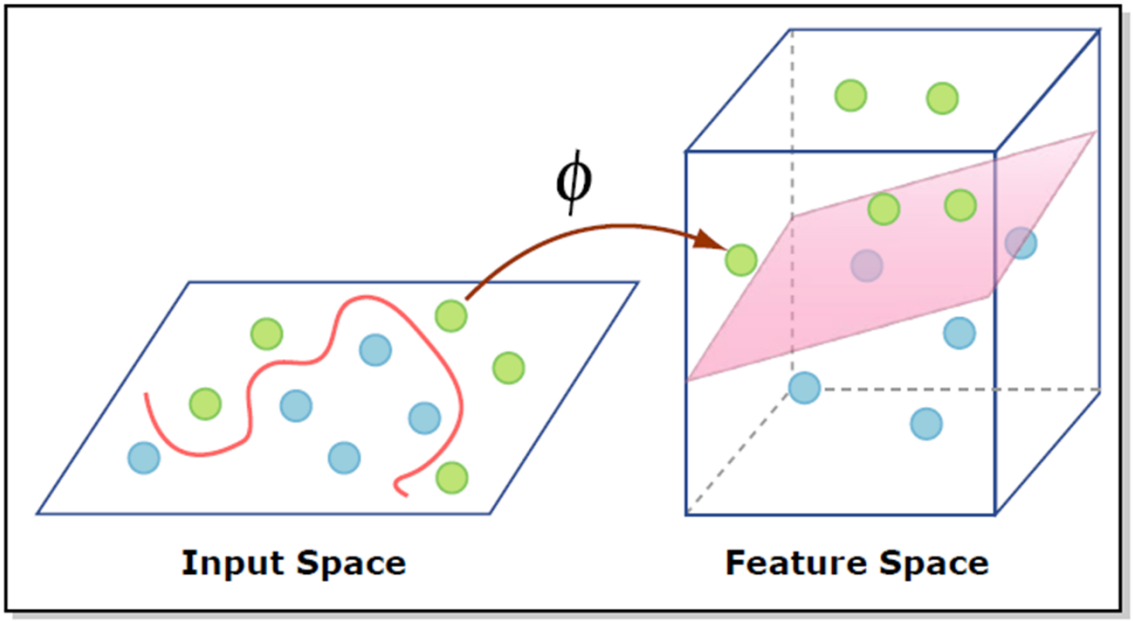
Separating Data

- We cannot separate some training sets with a linear decision boundary
- If we cannot separate the data, then we could try to transform the features.
  - Rearranging the features could separate the points
  - Adding features could provide space



Separating Data

- We cannot separate some training sets with a linear decision boundary
- If we cannot separate the data, then we could try to transform the features.
  - Rearranging the features could separate the points
  - Adding features could provide space



Separating Data

- For example, suppose we want to study real estate. Can we predict the sale of a house on a date?
- Eligibility of a house depends on attributes along with relationships between the attributes like $x^{(1)}$ $x^{(2)}$
- SVM would require calculating dot products of the training data

$$\mathbf{x} = \left[\underbrace{x^{(1)}}_{\text{house's list price}}, \underbrace{x^{(2)}}_{\text{estimated worth}}, \underbrace{x^{(3)}}_{\text{length of time on market}}, \underbrace{x^{(4)}}_{\text{in a good area}}, \ldots \right]$$

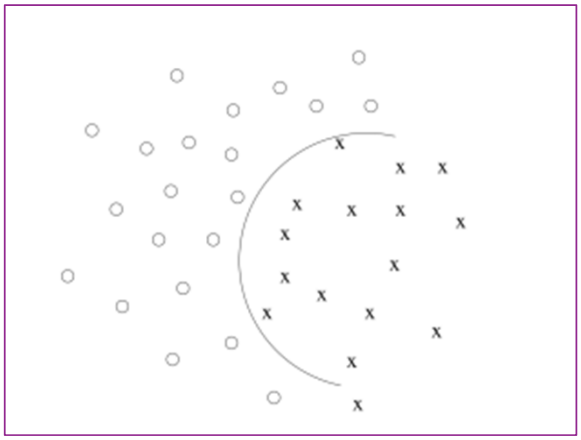
- Feature Transformation

$$[x^{(1)}, x^{(2)}] \rightarrow \mathbf{\Phi}\left([x^{(1)}, x^{(2)}]\right) = [x^{(1)2}, x^{(2)2}, x^{(1)}x^{(2)}]$$

- Two Dimensions to Three Dimensions with dot product

$$\mathbf{\Phi}(\mathbf{x})^T \mathbf{\Phi}(\mathbf{z}) = x^{(1)2}z^{(1)2} + x^{(2)2}z^{(2)2} + x^{(1)}x^{(2)}z^{(1)}z^{(2)}$$

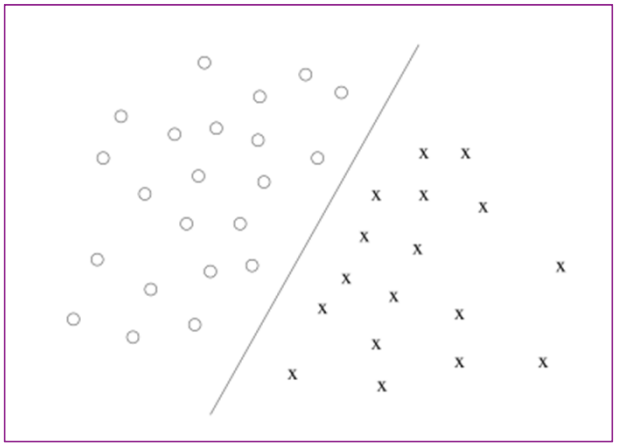
Separating Data



$$\mathbf{x} = \left[\underbrace{x^{(1)}}_{\text{house's list price}}, \underbrace{x^{(2)}}_{\text{estimated worth}}, \underbrace{x^{(3)}}_{\text{length of time on market}}, \underbrace{x^{(4)}}_{\text{in a good area}}, \ldots \right]$$

- Feature Transformation

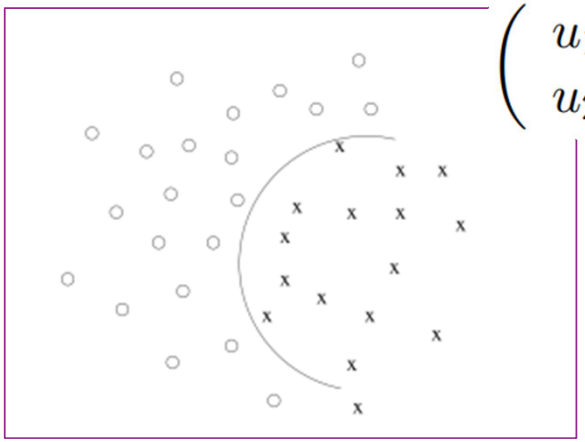
$$[x^{(1)}, x^{(2)}] \rightarrow \mathbf{\Phi}\left([x^{(1)}, x^{(2)}]\right) = [x^{(1)2}, x^{(2)2}, x^{(1)}x^{(2)}]$$

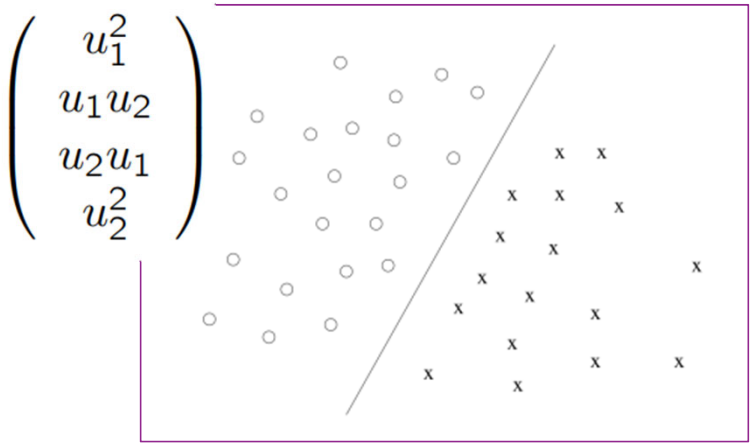


- Two Dimensions to Three Dimensions with dot product

$$\mathbf{\Phi}(\mathbf{x})^T \mathbf{\Phi}(\mathbf{z}) = x^{(1)2}z^{(1)2} + x^{(2)2}z^{(2)2} + x^{(1)}x^{(2)}z^{(1)}z^{(2)}$$

Separating Data





$$\begin{pmatrix} u_1^2 \\ u_1u_2 \\ u_2u_1 \\ u_2^2 \end{pmatrix} \cdot \begin{pmatrix} v_1^2 \\ v_1v_2 \\ v_2v_1 \\ v_2^2 \end{pmatrix} = u_1^2v_1^2 + 2u_1v_1u_2v_2 + u_2^2v_2^2 = (u_1v_1 + u_2v_2)^2 = (u.v)^2$$

- Feature Transformation

$$[x^{(1)}, x^{(2)}] \rightarrow \mathbf{\Phi}\left([x^{(1)}, x^{(2)}]\right) = \left[x^{(1)2}, x^{(2)2}, x^{(1)}x^{(2)}\right]$$

- Two Dimensions to Three Dimensions with dot product

$$\mathbf{\Phi}(\mathbf{x})^T\mathbf{\Phi}(\mathbf{z}) = x^{(1)2}z^{(1)2} + x^{(2)2}z^{(2)2} + x^{(1)}x^{(2)}z^{(1)}z^{(2)}$$

Inner Product

- Rather than use SVM with x coordinates we use them with Φ(x) coordinates
- For some algorithm such as SVM we the training data appears in dot products, we can replace the expressions with other inner products.
- Different inner products implicitly capture different feature transformations

Inner Product

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,k=1}^{m} \alpha_i \alpha_j y_i y_k \, \mathbf{x}_i^T \mathbf{x}_k$$

$$\text{s.t. } 0 \leq \alpha_i \leq C, i = 1, ..., m \text{ and } \sum_{i=1}^{m} \alpha_i y_i = 0$$

Replace $x^T z$ with $k(x, z)$ where k is inner product in some space corresponding to the feature transformation $\Phi(x)^T \Phi(z)$

Inner Products

- Inner products associate pairs of elements in feature space *X* with numbers
- We define inner products by three properties that generalize dot products

$$\langle u, v \rangle_{\mathbf{R}^n} = u^T v,$$

- Vector space equipped with inner product is called a Hilbert space

Example would be l_2 summable sequences with

$$\langle u, v \rangle_{\ell_2} = \sum_{i=1}^{\infty} u_i v_i$$

1. Symmetry

$$\langle u, v \rangle = \langle v, u \rangle \quad \forall u, v \in \mathcal{X}$$

2. Bilinearity

$$\langle \alpha u + \beta v, w \rangle = \alpha \langle u, w \rangle + \beta \langle v, w \rangle \quad \forall u, v, w \in \mathcal{X}, \forall \alpha, \beta \in \mathbf{R}$$

3. Strict Positive Definiteness

$$\langle u, u \rangle \geq 0 \quad \forall x \in \mathcal{X}$$

$$\langle u, u \rangle = 0 \iff u = 0$$

Moreover we need a technical condition that space does not have any holes.

Kernel Trick

- Suppose we take k(x,z) to be square of the dot product

$$k(\mathbf{x},\mathbf{z}) = \langle \mathbf{x},\mathbf{z}\rangle^2_{\mathbf{R}^n} = \left(\sum_{j=1}^{n} x^{(j)} z^{(j)}\right)^2 = \sum_{j=1}^{n}\sum_{k=1}^{n} x^{(j)} x^{(k)} z^{(j)} z^{(k)}$$

- Using an observation from Homework 4 we could deduce that k(x,z) is an inner product in some space corresponding to a feature transformation. However for n=2

$$\mathbf{\Phi}\left([x^{(1)}, x^{(2)}]\right) = \left[x^{(1)2}, x^{(2)2}, x^{(1)}x^{(2)}, x^{(2)}x^{(1)}\right]$$

give us

$$\mathbf{\Phi}(\mathbf{x})^T\mathbf{\Phi}(\mathbf{z}) \;=\; x^{(1)2}z^{(1)2} + x^{(2)2}z^{(2)2} + 2x^{(1)}x^{(2)}z^{(1)}z^{(2)} = \langle \mathbf{x},\mathbf{z}\rangle^2_{\mathbf{R}^2}$$

Kernel Trick

- We could add a constant c

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

- For n=3 we could take Φ to be

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

where c control weight on linear versus quadratic terms

Kernel Trick

- We could add a constant c

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

$$= \sum_{j=1}^{n}\sum_{\ell=1}^{n} x^{(j)}x^{(\ell)}z^{(j)}z^{(\ell)} + 2c\sum_{j=1}^{n} x^{(j)}z^{(j)} + c^2$$

$$= \sum_{j,\ell=1}^{n} (x^{(j)}x^{(\ell)})(z^{(j)}z^{(\ell)}) + \sum_{j=1}^{n} (\sqrt{2c}x^{(j)})(\sqrt{2c}z^{(j)}) + c^2,$$

We could take any exponent d in place of 2

How many features would we need for the kernel?

- For n=3 we could take Φ to be

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

where c control weight on linear versus quadratic terms

Dimension of Feature Space

- If we take a positive integer d for exponent

$$k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T\mathbf{z} + c)^d$$

then we would need

$\binom{n+d}{d}$ dimensions

- We can compute the number through “bars and stars” argument.

$\binom{r+3-1}{r}$

1	3
2	6
3	10
4	15
5	21
6	28
7	36
8	45
9	55

$\binom{r+30-1}{r}$

1	30
2	465
3	4960
4	40920
5	278256
6	1623160
7	8347680
8	38608020
9	163011640

Kernel Trick

Kernel Trick

- If the training data appears in the objective function and prediction function within inner products then we can swap out the linear kernel for another kernel
- Other kernel may correspond to a high-dimensional feature space.
- However once the kernel matrix is computed, the computational cost depends on number of data points, rather than the dimension of feature space.

If you have implemented your method in terms of a kernel, you can go from a kernel in a small feature space to a kernel in large feature space without refactoring your code.

Kernel Trick

Running time unaffected after you have computed the inner products

Kernel Trick

- If the training data appears in the objective function and prediction function within inner products then we can swap out the linear kernel for another kernel
- Other kernel may correspond to a high-dimensional feature space.
- However once the kernel matrix is computed, the computational cost depends on number of data points, rather than the dimension of feature space.

If you have implemented your method in terms of a kernel, you can go from a kernel in a small feature space to a kernel in large feature space without refactoring your code.

Gram Matrix

How do we ensure that the kernel corresponds to a feature transformation?

- Suppose we have points $\{x_1,\ldots,x_m\}$ that represent all possible training data
- For example, if we have bag of words encoding, then m = $2^{\#words}$
- Form a matrix with

  $k(x_i, x_j)$ in row i and column j

Note that inner products are symmetric. If the kernel comes from a feature transformation then the Gram matrix is symmetric

$$K = \begin{array}{c} \\ m \\ \vdots \\ 1 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & j & m \end{array} \\ \begin{bmatrix} & & \\ & k(x_i, x_j) & \\ & & \end{bmatrix} \end{array}$$

Gram Matrix

How do we ensure that the kernel corresponds to a feature transformation?

- By the singular value decomposition we have

$$K = V \Lambda V^T$$

for diagonal matrix Λ and orthogonal matrix consisting of eigenvectors V

- Here we use symmetry to have $V \Lambda V^T$ instead of

$U \Lambda V^T$ for some other U

Note that inner products are symmetric. If the kernel comes from a feature transformation then the Gram matrix is symmetric

$$K = \begin{array}{c} \\ m \\ i \\ 1 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & j & m \end{array} \\ \begin{bmatrix} & & \\ & k(x_i, x_j) & \\ & & \end{bmatrix} \end{array}$$

Gram Matrix

How do we ensure that the kernel corresponds to a feature transformation?

- By the singular value decomposition we have

$$K = V \Lambda V^T$$

for diagonal matrix Λ and orthogonal matrix consisting of eigenvectors V

- Here we use symmetry to have $V \Lambda V^T$ instead of

$U \Lambda V^T$ for some other U

- Assuming the entries λ of Λ are non-negative we can form the feature map

$$\Phi(x_i) = [\sqrt{\lambda_1} v_1^{(i)}, ..., \sqrt{\lambda_t} v_t^{(i)}, ..., \sqrt{\lambda_m} v_m^{(i)}]$$

- Therefore the dot product in R^m is

$$\langle \Phi(x_i), \Phi(x_j) \rangle_{\mathbf{R}^m} = \sum_{t=1}^{m} \lambda_t v_t^{(i)} v_t^{(j)}$$

$$= (\mathbf{V} \mathbf{\Lambda} \mathbf{V}')_{ij} = K_{ij} = k(x_i, x_j)$$

Gram Matrix

How do we ensure that the kernel corresponds to a feature transformation?

- Note that non-negative sign of the eigenvalues is important for the feature map
- Suppose we took

$$\mathbf{z} = \sum_{i=1}^{m} v_s^{(i)} \mathbf{\Phi}(x_i)$$

a combination of the transformed training data

- Assuming the entries λ_i of Λ are non-negative we can form the feature map

$$\mathbf{\Phi}(x_i) = [\sqrt{\lambda_1} v_1^{(i)}, ..., \sqrt{\lambda_t} v_t^{(i)}, ..., \sqrt{\lambda_m} v_m^{(i)}]$$

- Therefore the dot product in R^m is

$$\langle \mathbf{\Phi}(x_i), \mathbf{\Phi}(x_j) \rangle_{\mathbf{R}^m} = \sum_{t=1}^{m} \lambda_t v_t^{(i)} v_t^{(j)}$$

$$= (\mathbf{V}\mathbf{\Lambda}\mathbf{V}')_{ij} = K_{ij} = k(x_i, x_j)$$

Gram Matrix

How do we ensure that the kernel corresponds to a feature transformation?

- Note that non-negative sign of the eigenvalues is important for the feature map
- Suppose we took

$$\mathbf{z} = \sum_{i=1}^{m} v_s^{(i)} \mathbf{\Phi}(x_i)$$

a combination of the transformed training data

- We can calculate

$$\begin{aligned} \|\mathbf{z}\|_2^2 &= \langle \mathbf{z}, \mathbf{z} \rangle_{\mathbf{R}^m} = \sum_i \sum_j v_s^{(i)} \mathbf{\Phi}(x_i)^T \mathbf{\Phi}(x_j)\, v_s^{(j)} \\ &= \sum_i \sum_j v_s^{(i)} K_{ij} v_s^{(j)} \\ &= \mathbf{v}_s^T \mathbf{K} \mathbf{v}_s = \lambda_s < 0 \end{aligned}$$

- Therefore the sign must be non-negative

Exercise

Consider a set of vectors $S = \{x_1, \ldots, x_m\}$.

Let X denote the matrix whose rows are these vectors.

Form the Gram matrix $K = XX^T$. Show that knowing K is equivalent to knowing the set of pairwise distances among the vectors in S as well as the vector lengths.

Note that the distance between x and y is given by

$$d(x, y) = \|x - y\|$$

and the norm of a vector x is defined as

$$\|x\| = \sqrt{\langle x, x \rangle} = \sqrt{x^T x}$$

Exercise

Consider a set of vectors $S = \{x_1, \ldots, x_m\}$.

Let X denote the matrix whose rows are these vectors.

Form the Gram matrix $K = XX^T$. Show that knowing K is equivalent to knowing the set of pairwise distances among the vectors in S as well as the vector lengths.

Note that the distance between x and y is given by

$$d(x, y) = \|x - y\|$$

and the norm of a vector x is defined as

$$\|x\| = \sqrt{\langle x, x \rangle} = \sqrt{x^T x}$$

$$\begin{aligned} d(x,y) &:= \sqrt{\langle x-y, x-y \rangle} \\ &= \sqrt{\langle x,x \rangle + \langle y,y \rangle - 2\langle x,y \rangle} \end{aligned}$$

Mercer Kernels

- Since we want the kernels to correspond to feature transformations we will look at Mercer kernels satisfying the conditions
  - k is symmetric meaning k(x,y) = k(y,x)
  - k determines a positive semi-definite Gram matrix for any points $\{x_1,\ldots,x_m\}$ in the space
- Another way to show that K is positive semi-definite is to check that for all vectors v, we have

$$v^T K v \geq 0$$

- The condition is equivalent to all eigenvalues being non-negative
- sssss

Mercer Kernels

- For example, if m = 2 then we have the Gram matrix

$$\mathbf{K} = \begin{pmatrix} k(u,u) & k(u,v) \\ k(v,u) & k(v,v) \end{pmatrix}$$

- Take v to be

$$\begin{bmatrix} k(v,v) \\ -k(u,v) \end{bmatrix}$$

- Since $0 \leq v^T K v$ we have

$$0 \leq \; [k(v,v)k(u,u) - k(u,v)^2]k(v,v)$$

- Therefore we have

$$\boxed{k(v,v)k(u,u) \geq k(u,v)^2}$$

the Cauchy-Schwarz inequality

Identifying Kernels

How can we incorporate kernels particularly Gram matrices in to Support Vector Machines?

- For Dual Form of Soft Margin Support Vector Machine we have objective

$$\max_{\alpha \in \mathbf{R}^n} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_j^T x_i$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\alpha_i \in \left[0, \frac{c}{n}\right] \quad i = 1, \ldots, n.$$

- For optimal dual variables we have

$$w^* = \sum_{i=1}^{n} \alpha_i^* y_i x_i$$

and

$$x^T w^* = \sum_{i=1}^{n} \alpha_i^* y_i x_i^T x.$$

- So the algorithm is kernelized meaning that training data appears within inner products

Identifying Kernels

How can we incorporate kernels particularly Gram matrices in to Support Vector Machines?

- Suppose we have a weight in the span of the data

$$w = \sum_{i=1}^{n} \alpha_i x_i$$

where the coefficients are the dual variables

- We can calculate the l_2 norm squared by

$$\begin{aligned} \|w\|^2 &= \langle w, w \rangle \\ &= \left\langle \sum_{i=1}^{n} \alpha_i x_i, \sum_{j=1}^{n} \alpha_j x_j \right\rangle \\ &= \sum_{i,j=1}^{n} \alpha_i \alpha_j \langle x_i, x_j \rangle \\ &= \alpha^T K \alpha. \end{aligned}$$

Identifying Kernels

How can we incorporate kernels particularly Gram matrices in to Support Vector Machines?

- Moreover we can compute the inner product

$$\begin{aligned}\langle w, x_j \rangle &= \left\langle \sum_{i=1}^{n} \alpha_i x_i, x_j \right\rangle \\ &= \sum_{i=1}^{n} \alpha_i \langle x_i, x_j \rangle\end{aligned}$$

- We can calculate the l_2 norm squared by

$$\begin{aligned}\|w\|^2 &= \langle w, w \rangle \\ &= \left\langle \sum_{i=1}^{n} \alpha_i x_i, \sum_{j=1}^{n} \alpha_j x_j \right\rangle \\ &= \sum_{i,j=1}^{n} \alpha_i \alpha_j \langle x_i, x_j \rangle \\ &= \alpha^T K \alpha.\end{aligned}$$

Identifying Kernels

How can we incorporate kernels particularly Gram matrices in to Support Vector Machines?

- Moreover we can compute the inner product

$$\begin{aligned}\langle w, x_j \rangle &= \left\langle \sum_{i=1}^{n} \alpha_i x_i, x_j \right\rangle \\ &= \sum_{i=1}^{n} \alpha_i \langle x_i, x_j \rangle\end{aligned}$$

- Putting it together we can compute the score for each point in the training data

$$\begin{aligned}s\left(\sum_{i=1}^{n} \alpha_i x_i\right) &= \begin{pmatrix} \sum_{i=1}^{n} \alpha_i \langle x_i, x_1 \rangle \\ \vdots \\ \sum_{i=1}^{n} \alpha_i \langle x_i, x_n \rangle \end{pmatrix} \\ &= \begin{pmatrix} \alpha_1 \langle x_1, x_1 \rangle + \cdots + \alpha_n \langle x_n, x_1 \rangle \\ \vdots \\ \alpha_1 \langle x_1, x_n \rangle + \cdots + \alpha_n \langle x_n, x_n \rangle \end{pmatrix} \\ &= \begin{pmatrix} \langle x_1, x_1 \rangle & \cdots & \langle x_1, x_n \rangle \\ \vdots & \ddots & \cdots \\ \langle x_n, x_1 \rangle & \cdots & \langle x_n, x_n \rangle \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix} = K\alpha\end{aligned}$$

Identifying Kernels

How can we incorporate kernels particularly Gram matrices in to Support Vector Machines?

- For Primal Form of Soft Margin Support Vector Machine we have objective

$$\min_{w \in \mathbf{R}^d} \frac{1}{2}\|w\|^2 + \frac{c}{n}\sum_{i=1}^{n} \max\left(0, 1 - y_i w^T x_i\right)$$

- Substituting for w

$$w = \sum_{i=1}^{n} \alpha_i x_i$$

we obtain

$$\min_{\alpha \in \mathbf{R}^n} \frac{1}{2}\alpha^T K \alpha + \frac{c}{n}\sum_{i=1}^{n} \max\left(0, 1 - y_i \left(K\alpha\right)_i\right)$$

with prediction given by

$$x^T w^* = \sum_{i=1}^{n} \alpha_i^* x_i^T x$$

Representer Theorem

- Suppose we want to generalize from Support Vector Machines to other algorithms.
- For example, how could we identify kernels in Ridge Regression just by the form of the objective function?
- Remember the objective function should include both empirical risk and regularization

- We can take a general prediction function to be

$$s(w) = \begin{pmatrix} \langle w, x_1 \rangle \\ \vdots \\ \langle w, x_n \rangle \end{pmatrix}$$

- We can express the objective function as

$$J(w) = R(\|w\|) + L(s(w))$$

where L takes column vector as input

Representer Theorem

Theorem (Representer Theorem)

Let

$$J(w) = R(\|w\|) + L(\langle w, x_1 \rangle, \ldots, \langle w, x_n \rangle),$$

where

- $w, x_1, \ldots, x_n \in \mathcal{H}$ *for some Hilbert space* $\mathcal{H}$. *(We typically have* $\mathcal{H} = \mathbf{R}^d$.*)*
- $\|\cdot\|$ *is the norm corresponding to the inner product of* $\mathcal{H}$. *(i.e.* $\|w\| = \sqrt{\langle w, w \rangle}$*)*
- $R : [0, \infty) \to \mathbf{R}$ *is nondecreasing (**Regularization term**), and*
- $L : \mathbf{R}^n \to \mathbf{R}$ *is arbitrary (**Loss term**).*

If $J(w)$ *has a minimizer, then it* ***has a minimizer of the form*** $w^* = \sum_{i=1}^{n} \alpha_i x_i$.

Representer Theorem

- Representer Theorem tells us to look in the span of the data
- If we switch to studying the coefficients multiplying the training data then we can change objective.

$$w^* = \operatorname*{arg\,min}_{w \in \operatorname{span}(x_1, \ldots, x_n)} R(\|w\|) + L(\langle w, x_1 \rangle, \ldots, \langle w, x_n \rangle)$$

$$\alpha^* = \operatorname*{arg\,min}_{\alpha \in \mathbf{R}^n} R\left(\left\|\sum_{i=1}^n \alpha_i x_i\right\|\right) + L\left(\left\langle \sum_{i=1}^n \alpha_i x_i, x_1 \right\rangle, \ldots, \left\langle \sum_{i=1}^n \alpha_i x_i, x_n \right\rangle\right)$$

Representer Theorem

- Representer Theorem tells us to look in the span of the data
- If we switch to studying the coefficients multiplying the training data then we can change objective.
- Information contained in Gram matrix K

$$\alpha^* = \underset{\alpha \in \mathbf{R}^n}{\arg\min}\, R\left(\left\|\sum_{i=1}^{n} \alpha_i x_i\right\|\right) + L\left(\left\langle \sum_{i=1}^{n} \alpha_i x_i, x_1 \right\rangle, \ldots, \left\langle \sum_{i=1}^{n} \alpha_i x_i, x_n \right\rangle\right)$$

- For wide dataset the Gram matrix will be computationally sensible approach

$$\begin{aligned} J_0(\alpha) &= R\left(\left\|\sum_{i=1}^{n} \alpha_i x_i\right\|\right) + L\left(s\left(\sum_{i=1}^{n} \alpha_i x_i\right)\right) \\ &= R\left(\sqrt{\alpha^T K \alpha}\right) + L(K\alpha), \end{aligned}$$

Examples of Kernels

- Consider feature encoding for strings of characters in some alphabet $\{A, R, N, D, C, E, Q, G, H, I, L, K, M, F, P, S, T, W, Y, V\}$
- How should we relate the following strings?

Use data structure called trie to efficiently compute common substrings

```
IPTSALVKETLALLSTHRTLLIANETLRIPVPVHKNHQLCTEEIFQGIGTLESQTVQGGTV
ERLFKNLSLIKKYIDGQKKKCGEERRRVNQFLDYLQEFLGVMNTEWI

PHRRDLCSRSIWLARKIRSDLTALTESYVKHQGLWSELTEAERLQENLQAYRTFHVLLA
RLLEDQQVHFTPTEGDFHQAIHTLLLQVAAFAYQIEELMILLEYKIPRNEADGMLFEKK
LWGLKVLQELSQWTVRSIHDLRFISSHQTGIP
```

- We can associate weights and functions with each substring to form a generalization of bag-of-words

$$\kappa(x, x') = \sum_{s \in \mathcal{A}^*} w_s \phi_s(x) \phi_s(x')$$

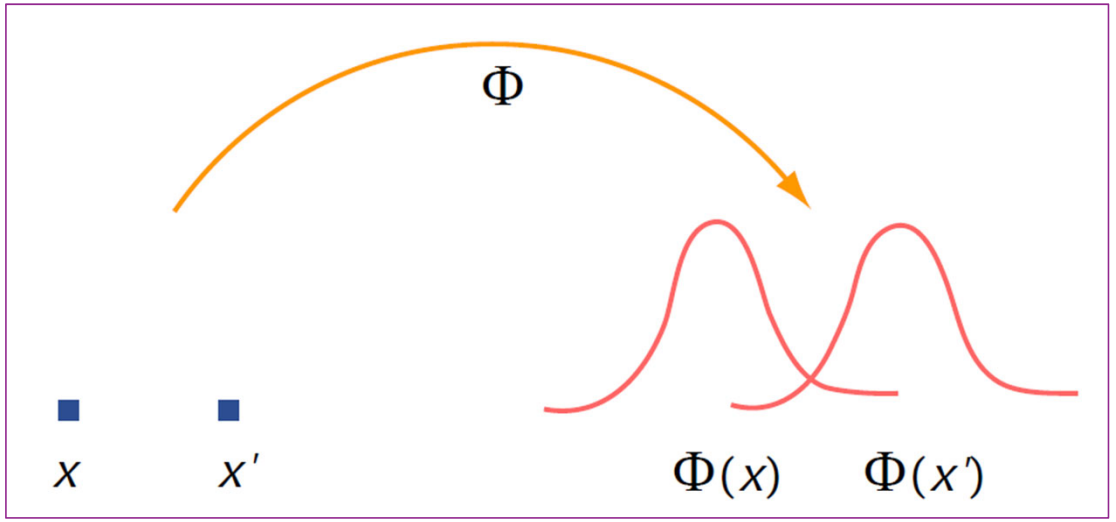
Examples of Kernels

- Radial basis function take the form

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

- The score associated with a radial basis function is

$$y \leftarrow \text{sign}\left[\sum_i \alpha_i y_i \exp\left(-\frac{\|\vec{x} - \vec{x}_i\|_2^2}{2\sigma^2}\right) + b\right]$$



Examples of Kernels

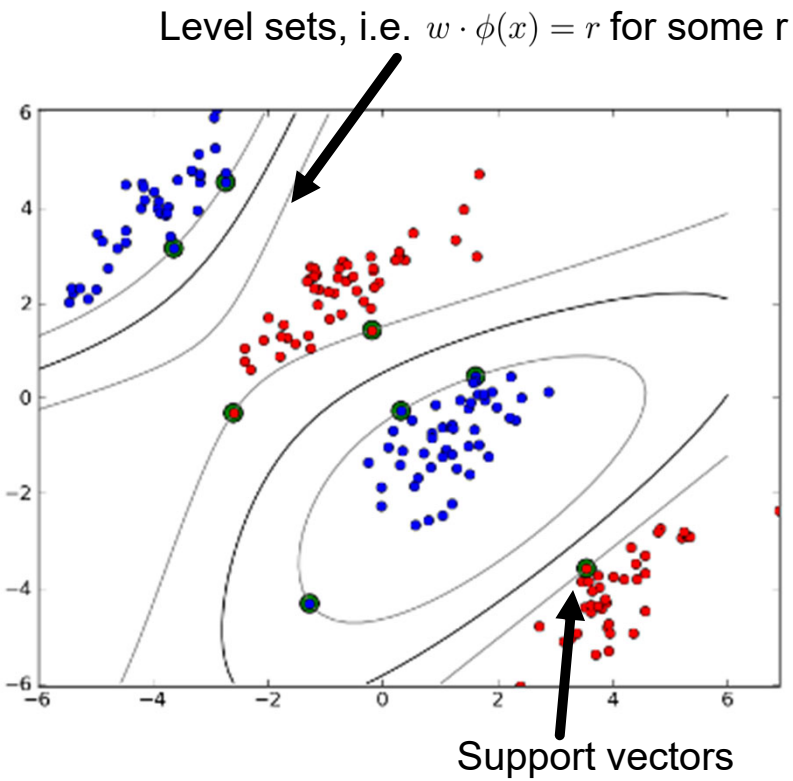
- Radial basis function take the form

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

- The score associated with a radial basis function is

$$y \leftarrow \text{sign}\left[\sum_i \alpha_i y_i \exp\left(-\frac{\|\vec{x} - \vec{x}_i\|_2^2}{2\sigma^2}\right) + b\right]$$

- How could you prove it's a kernel



Examples of Kernels

- Radial basis function take the form

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

- The score associated with a radial basis function is

$$y \leftarrow \text{sign}\left[\sum_i \alpha_i y_i \exp\left(-\frac{||\vec{x} - \vec{x}_i||_2^2}{2\sigma^2}\right) + b\right]$$

- How could you prove it's a kernel

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) $k(\mathbf{x}, \mathbf{v}) = f(\mathbf{x}) f(\mathbf{v}) k_a(\mathbf{x}, \mathbf{v})$	$\phi(\mathbf{x}) = f(\mathbf{x})\phi_a(\mathbf{x})$

- How could you prove it's a kernel

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

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):

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

Exercise

Show that for any kernels k_1 and k_2 the product

$$k(x, x') = k_1(x,x')\, k_2(x,x')$$

is a kernel

Try to take the outer product of the feature transformations

$$\phi(x) = \phi_1(x)\,[\phi_2(x)]^T$$

Exercise

Show that for any kernels k_1 and k_2 the product

$$k(x, x') = k_1(x,x')\, k_2(x,x')$$

is a kernel

Try to take the outer product of the feature transformations

$$\phi(x) = \phi_1(x)\,[\phi_2(x)]^T$$

$$\begin{aligned}
\langle \phi(x), \phi(x') \rangle &= \sum_{i,j} \phi(x)\phi(x') \\
&= \sum_{i,j} \left[\phi_1(x)\,[\phi_2(x)]^T\right]_{ij} \left[\phi_1(x')\,[\phi_2(x')]^T\right]_{ij} \\
&= \sum_{i,j} [\phi_1(x)]_i\, [\phi_2(x)]_j\, [\phi_1(x')]_i\, [\phi_2(x')]_j \\
&= \left(\sum_i [\phi_1(x)]_i\, [\phi_1(x')]_i\right) \left(\sum_j [\phi_2(x)]_j\, [\phi_2(x')]_j\right) \\
&= k_1(x,x')k_2(x,x')
\end{aligned}$$

Subgradients

- Convex applies to both sets and functions
- Set C is convex if for any x_1 and x_2 in C we have

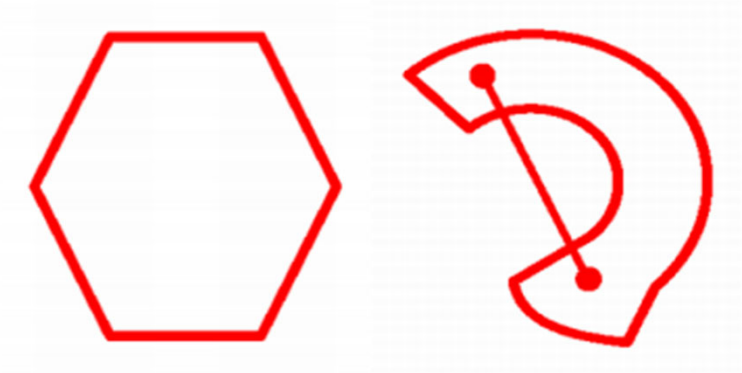
$$\theta x_1 + (1-\theta)x_2 \in C$$

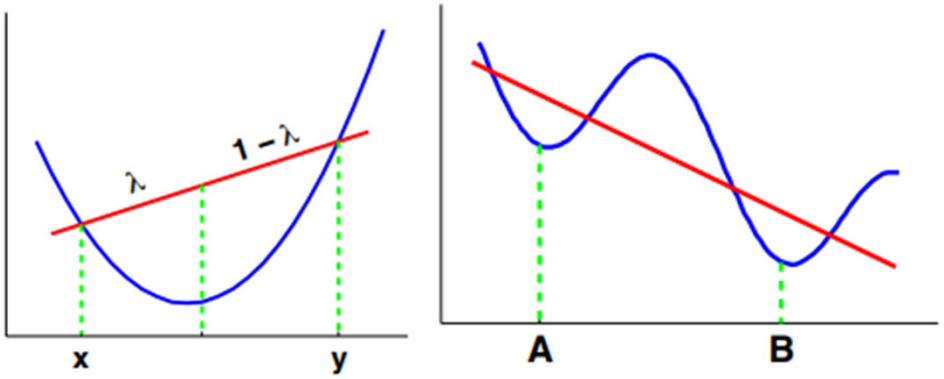
for any $0 \leqslant \theta \leqslant 1$

- Function $f : \mathbf{R}^n \to \mathbf{R}$

is convex if for any x,y and $0 \leqslant \theta \leqslant 1$ we have

$$f(\theta x + (1-\theta)y) \leqslant \theta f(x) + (1-\theta) f(y)$$





Subgradients

- Suppose $f : \mathbf{R}^d \to \mathbf{R}$ is differentiable. While convex functions are not linear functions, they behave like linear functions in a one-sided sense. The Taylor expansion near x is

$$f(y) \approx f(x) + \nabla f(x)^T (y - x)$$

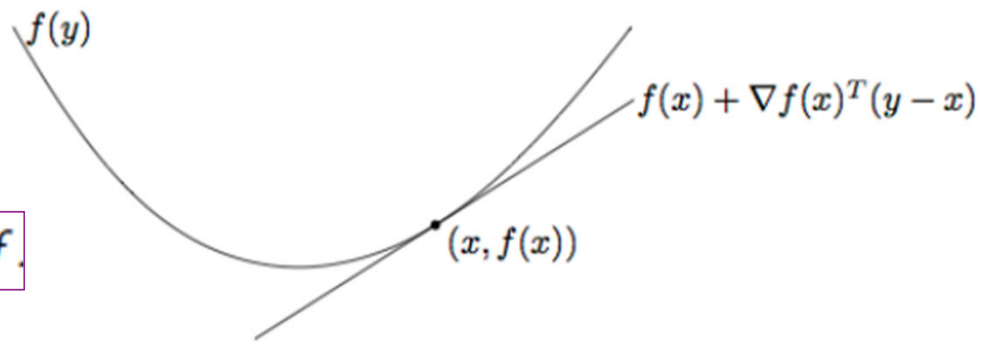
- By convexity we have

$$f(y) \geqslant f(x) + \nabla f(x)^T (y - x)$$

- Therefore the linear approximation

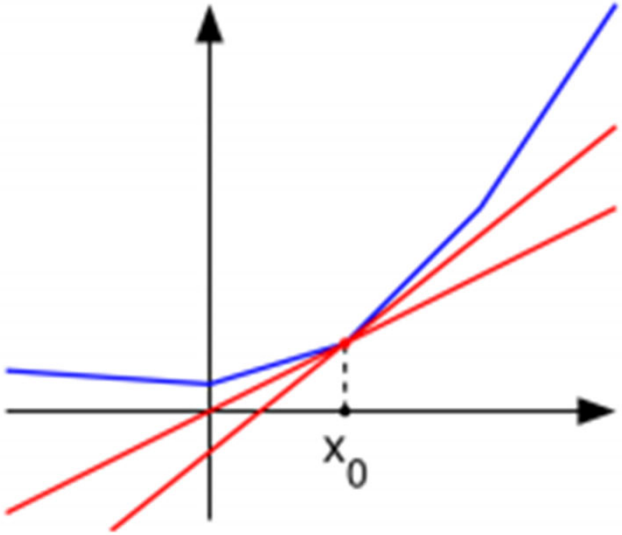
  near x determined by the gradient

  is a global under-estimator of f



If $\nabla f(x) = 0$ then x is a global minimizer of f.

Subgradients



A vector $g \in \mathbf{R}^d$ is a **subgradient** of $f : \mathbf{R}^d \to \mathbf{R}$ at x if for all z,

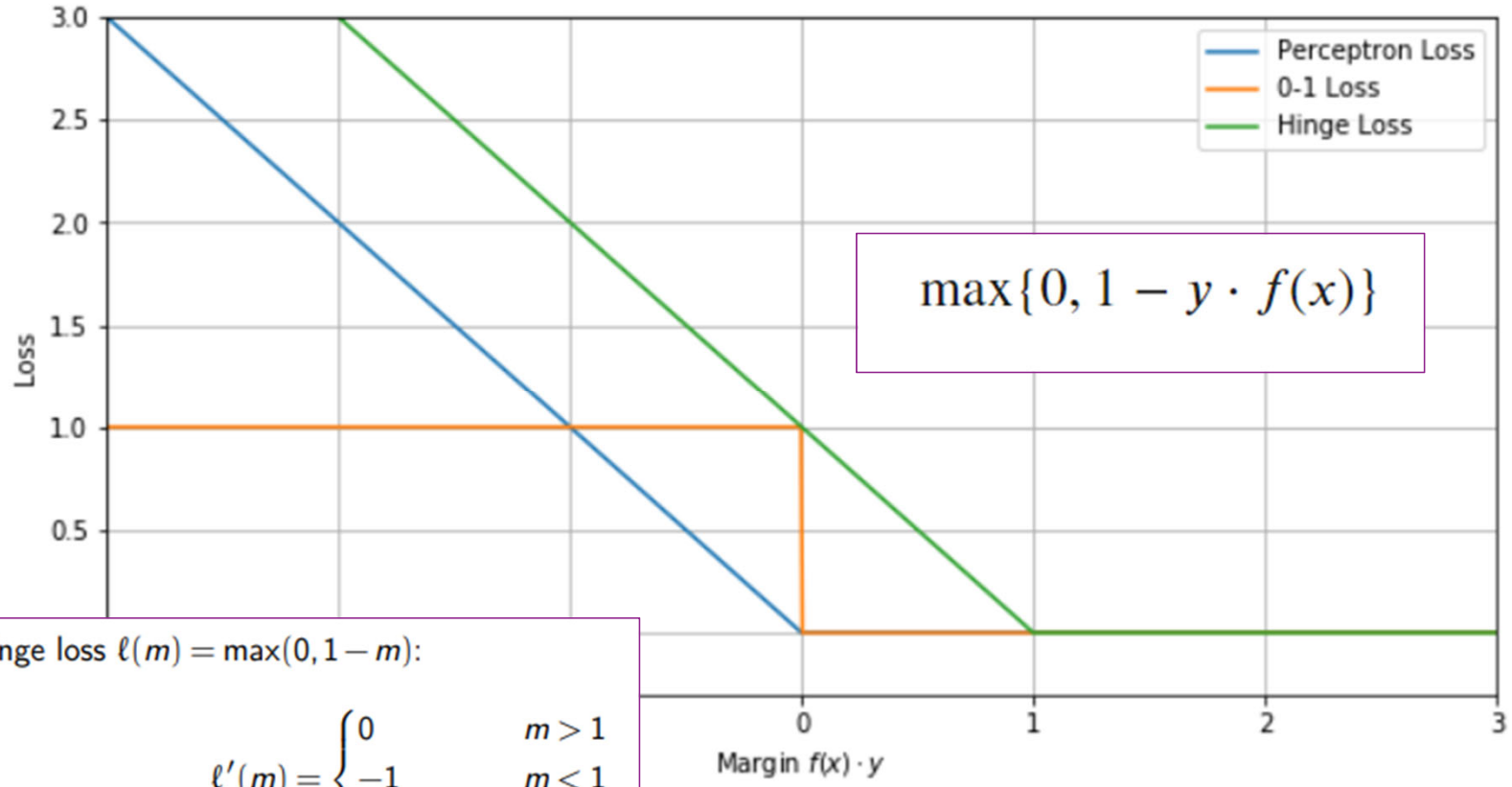
$$f(z) \geqslant f(x) + g^T(z - x).$$

f is **subdifferentiable** at x if $\exists$ at least one subgradient at x.
The set of all subgradients at x is called the **subdifferential:** $\partial f(x)$

Properties

- f is convex and differentiable $\implies \partial f(x) = \{\nabla f(x)\}$.
- Any point x, there can be 0, 1, or infinitely many subgradients.
- $\partial f(x) = \emptyset \implies f$ is not convex.

Subgradients

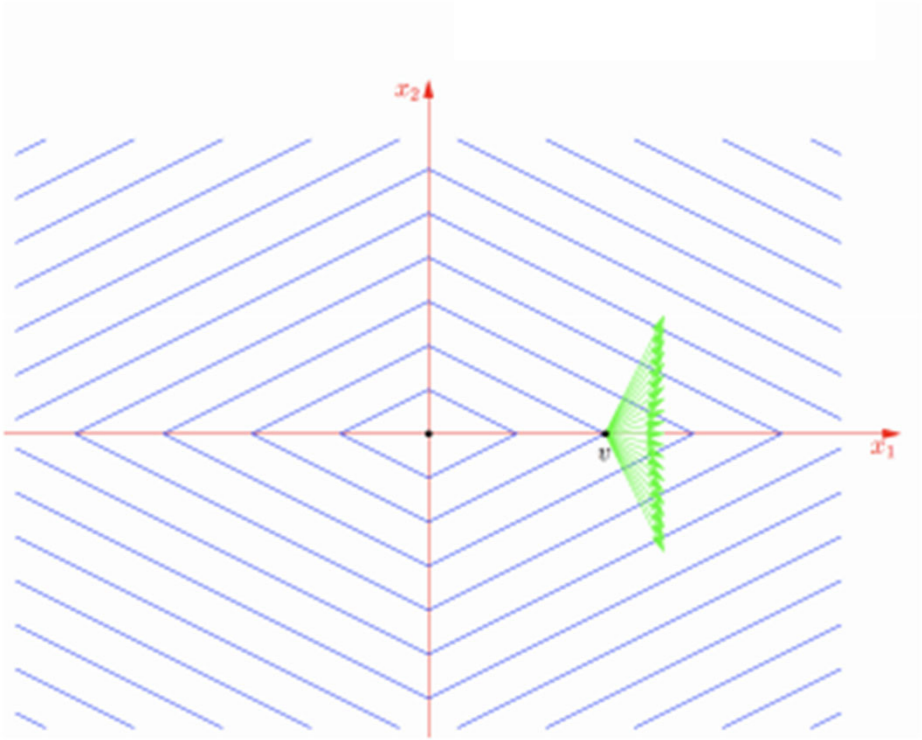


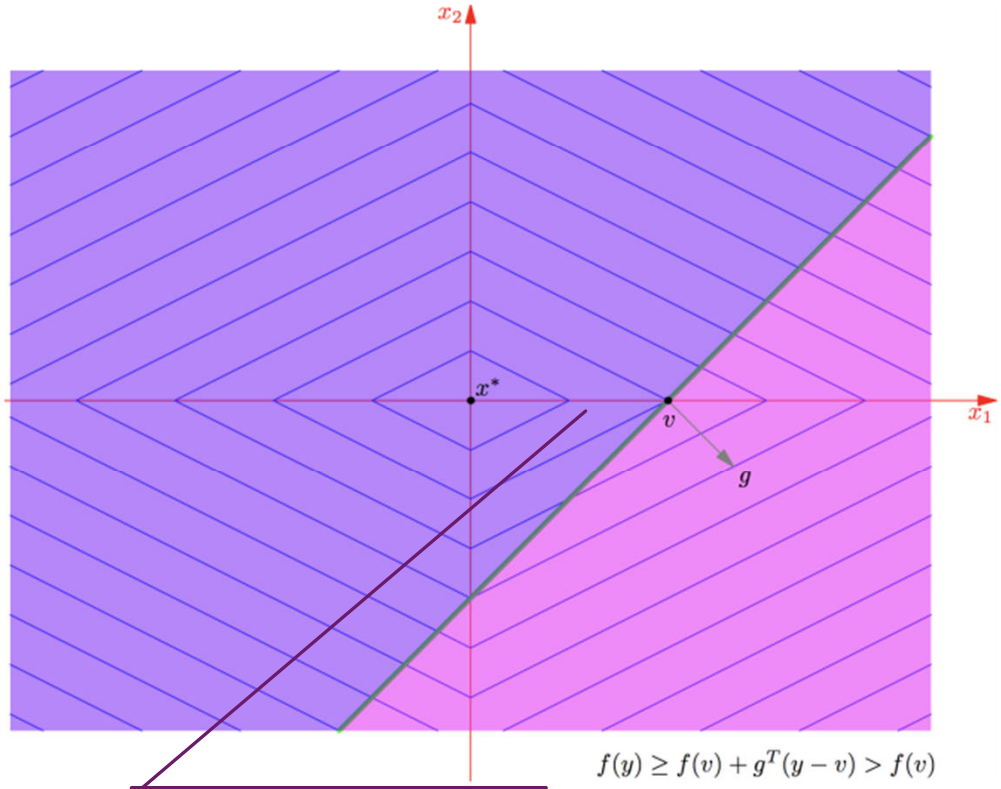
$$\max\{0, 1 - y \cdot f(x)\}$$

Derivative of hinge loss $\ell(m) = \max(0, 1-m)$:

$$\ell'(m) = \begin{cases} 0 & m > 1 \\ -1 & m < 1 \\ \text{undefined} & m = 1 \end{cases}$$

Subgradient Descent





-*g* is not a descent direction…the function might not decrease

Pegasos Algorithm

- **3 datasets** (provided by Joachims)
  - Reuters CCAT (800K examples, 47k features)
  - Physics ArXiv (62k examples, 100k features)
  - Covertype (581k examples, 54 features)

Training Time (in seconds):

	Pegasos	SVM-Perf	SVM-Light
Reuters	**2**	77	20,075
Covertype	**6**	85	25,514
Astro-Physics	**2**	5	80

Summary

- Support Vector Machines
  - Sparsity
  - Variants
- Kernels
  - Representer Theorem
- Training Support Vector Machines
  - Subgradients

Goals

- Could we use soft margin SVM with training data consisting of the same labels?
- What are linear, polynomial and radial basis function kernels?
- Why are SVM and Ridge Regression weights in the span of the data?
- How can be differentiate a function with corners?