ECE M146 Introduction to Machine Learning

Lecture 8 - Spring 2021

Prof. Lara Dolecek

ECE Department, UCLA

Today's Lecture

Recap:

- Methods for classification; perceptron

New topic:

- Support Vector Machines (SVM)

Methods for linear classification we have already learnt about

- Perceptron and logistic regression: both operate on $w^T x$
- Recall perceptron: it stops as soon as it finds a separating hyperplane, but this might be very close to a point, likely causing very high misclassification rate on the unseen points

- Next: another method for binary classification

Today's Lecture

Recap:

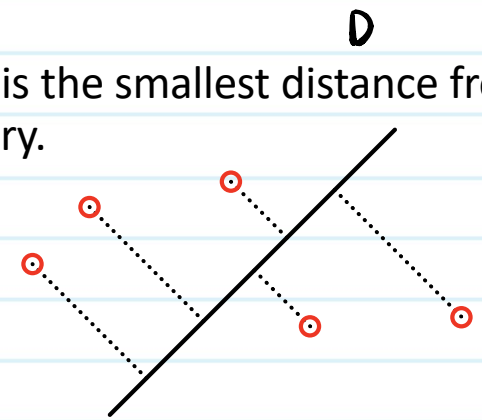
- Methods for classification; perceptron

New topic:

- Support Vector Machines (SVM)

Margin

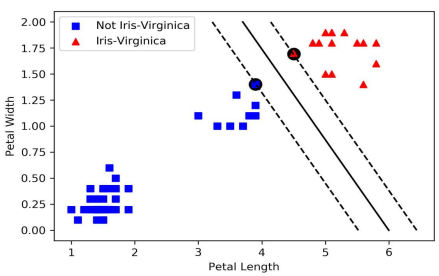
- Margin is the smallest distance from any training point to the decision boundary.
- Picture:

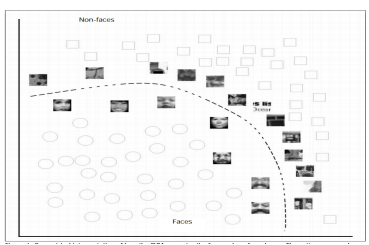


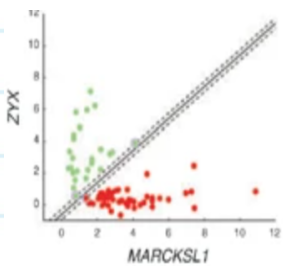
- Goal in SVM is to **maximize the margin**.
- Note that when the margin is maximized, there will be at least one point on each side (otherwise, can shift the margin more)

Applications

- Botanical classification (iris data set); file & text classification based on bag of words.
- Face vs. non face classification
- Cancerous vs. non-cancerous tumor based on gene expression







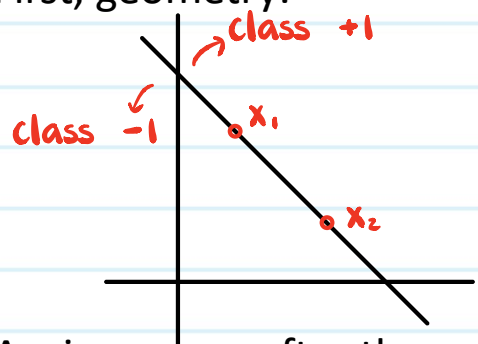
https://www-cdf.fnal.gov/physics/statistics/recommendations/svm/svm_ieee.pdf

https://www.nature.com/articles/nbt1206-1565

https://predictiveprogrammer.com/support-vector-machines-svms-a-must-have-ml-algorithm/

Set-up

- First, geometry:



$$y(x) = w^T x + b$$

$$y(x_1) = 0 \qquad w^T x_1 + b = 0$$

$$y(x_2) = 0 \qquad w^T x_2 + b = 0$$

$$w^T (x_1 - x_2) = 0$$

line

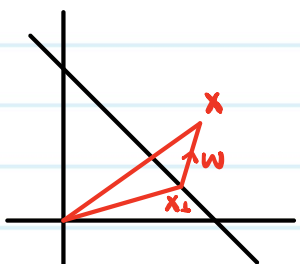
- Again, we are after the vector w perpendicular to the decision boundary.
- What is new is the new optimization rule to choose the best w.

Set-up

$w \perp D$

r: magnitude in direction of w

- Consider point x. Decompose it as follows:



$$x = x_\perp + r\frac{w}{\|w\|}$$

$x_\perp \in D$ in direction of w

- Note that the first term lies on the decision surface and the second term is in the direction of w.
- This decomposition is of course possible. What is interesting is what happens next:

Set-up

- Consider: $\underbrace{w^T x + b}_{y(x)} = \underbrace{w^T x_\perp + b}_{y(x_\perp) = 0} + r\frac{w}{\|w\|}$

$$r = \frac{y(x)}{\|w\|}$$ signed distance

$$y(x) = r \cdot \|w\|$$

- Here r is the distance of point x to the decision boundary $\mathcal{D}$. It can be positive or negative.
- We also introduce the unsigned distance: $\frac{t_n \cdot y(x_n)}{\|w\|}$

for pt x_n, $t_n \in \{-1, +1\}$

Optimization problem

- We optimize the vector w and the intercept term b (we didn't use dimensionality enlargement here) to maximize the unsigned distance:

$$\frac{t_n(w^T x_n + b)}{\|w\|}$$

- Smallest distance to the decision boundary over all N points:

$$\min_n \frac{t_n(w^T x_n + b)}{\|w\|}$$

- Thus, the optimization problem is:

$$\underset{w,b}{\operatorname{argmax}} \frac{1}{\|w\|} \min_n t_n(w^T x_n + b)$$

Convenient rescaling

- It is convenient to rescale parameters w and b as follows:

$$w \rightarrow k \cdot w \qquad b \rightarrow k \cdot b \qquad k > 0$$

- Note that the unsigned distance is unchanged:

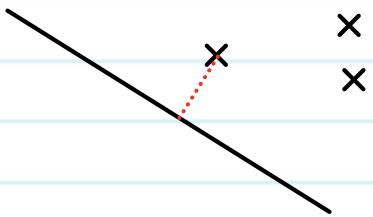
$$\frac{t_n y(x_n)}{\|w\|} \rightarrow \frac{t_n (k \cdot w^T x_n + k \cdot b)}{\|k \cdot w\|} = \frac{t_n (w^T x_n + b)}{\|w\|}$$

$$y(x_n) = w^T x_n + b$$

Convenient rescaling

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

- Consider a point x_n^* closest to the boundary:



$t_n(w^T x_n + b) = c \cdot \frac{1}{c}$

$t_n(kw^T x_n + kb) = 1$

- Key insight: All points then satisfy:

$t_n(kw^T x_n + kb) \geq 1 \quad \forall\, n,\ 1 \leq n \leq N$

Active and inactive points

- Points that satisfy the preceding inequality with equality are called **active points**, and all the rest are called **inactive points**.

- Note that for the optimized decision boundary, there will be at least one active point on each side (i.e., in each class)

Reformulation of the optimization problem

- We so far have: $\underset{w,b}{\operatorname{argmax}} \frac{1}{\|w\|} \min_n \underbrace{t_n(w^T x_n + b)}_{d_n \geq 1}$
- Which we can also write as: $\underset{w,b}{\operatorname{argmax}} \frac{1}{\|w\|} \min_n d_n$
- Equivalent to: $\underset{w,b}{\operatorname{argmax}} \frac{1}{\|w\|}$ subject to $t_n(w^T x_n + b) \geq 1 \quad \forall n$

which is $\sim$ $\underset{w,b}{\operatorname{argmin}} \frac{1}{2}\|w\|^2$ subject to $t_n(w^T x_n + b) \geq 1 \quad \forall n$

Reformulation of the optimization problem

- The last expression we had

$$\underset{w,b}{\operatorname{argmin}} \frac{1}{2}\|w\|^2 \quad \text{subject to } t_n(w^T x_n + b) \geq 1 \quad \forall n$$

- This is an instance of a quadratic program: minimize quadratic function subject to linear constraints.
- Example of a constrained optimization problem.

Lagrangian

- Next, we are going to convert a constrained optimization problem into an unconstrained optimization problem (a very general approach).
- We need a **Lagrangian**

$$L(w, b, a) = \frac{1}{2}\|w\|^2 - \sum_{n=1}^{N} a_n [t_n (w^T x_n - b) - 1]$$

- Terms a_n are called **Lagrange multipliers.**

≥ 0

a is vector of a_n's

New formulation

- Repeat Lagrangian

$$L = \frac{1}{2}\|w\|^2 + \sum_{n=1}^{N} a_n [1 - t_n(w^T x_n - b)]$$

- Interpretation: recall we want $t_n(w^T x_n - b) \geq 1$

if $t_n(w^T x_n - b) < 1$ at some (x_n, t_n):

$1 - t_n(w^T x_n - b) > 0$

otherwise $1 - t_n(w^T x_n - b) \leq 1$

Min-max

- Min-max of the Lagrangian:

$$\min_{w,b} \max_{a_n} L$$

- Why like this ?
- Inner maximization works against us.
- When the inequality condition at some x_n is violated, we can just set a_n to $+\infty$. Then the min(∞) is still ∞.

$$1 - t_n(w^T x - b) > 0 \quad \Rightarrow \quad a_n = \infty$$

Min-max

- Min-max of the Lagrangian:

$$\min_{n,b} \max_{a_n} L$$

- Why like this ?
- Inner maximization works against us.
- When the inequality condition is not violated, i.e., $1-t_n(w^T x_n+b)$ is at most 0, the best a_n can do is to be 0.

$$1 - t_n(w^T x - b) \leq 0 \quad \Rightarrow \quad a_n [1 - t_n(w^T x_n - b)] \leq 0$$

Set derivatives to 0.

- Set the derivative with respect to b of $\mathcal{L}$ to 0:

$$\frac{\partial L}{\partial b} = \frac{\partial}{\partial b}\left[\frac{1}{2}\|w\|^2 + \sum_{n=1}^{N} a_n[1 - t_n(w^T x_n - b)]\right] = 0$$

$$\Rightarrow \sum_{n=1}^{N} a_n t_n = 0$$

Set derivatives to 0.

- Set the derivative with respect to w of $\mathcal{L}$ to 0:

$$\frac{\partial L}{\partial w} = \frac{\partial}{\partial w}\left[\frac{1}{2}\|w\|^2 + \sum_{n=1}^{N} a_n[1 - t_n(w^T x_n - b)]\right] = 0$$

$$= w - \sum_{n=1}^{N} a_n t_n x_n$$

$$w = \sum_{n=1}^{N} a_n t_n x_n$$

Math, continued

- Substitute this w back into $\mathcal{L}$.

$$L = \frac{1}{2}\|w\|^2 + \sum_{n=1}^{N} a_n[1 - t_n(w^T x_n - b)]$$

$$= \frac{1}{2}\|\sum_{n=1}^{N} a_n t_n x_n\|^2 + \sum_{n=1}^{N} a_n[1 - t_n((\sum_{j=1}^{N} a_j t_j x_j)^T x_n + b)]$$

- Next, simplify this expression.

More math

- Go term by term.
- 1st term: $\frac{1}{2}\left\|\sum_{n=1}^{N} a_n t_n \mathbf{x}_n\right\|^2 = \frac{1}{2}\sum_{n=1}^{N} a_n^2 t_n^2 \|\mathbf{x}_n\|^2 + \frac{1}{2}\sum_{n=1}^{N}\sum_{\substack{k=1 \\ k\neq n}}^{N} a_n t_n a_k t_k \mathbf{x}_n^T \mathbf{x}_k$

$\|\mathbf{x}+\mathbf{y}\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 + 2\mathbf{x}^T\mathbf{y}$

$t_n^2 = 1$

More math

- Go term by term.
- 2nd term:

$$\sum_{n=1}^{N} a_n - \sum_{n=1}^{N} a_n t_n \left(\sum_{j=1}^{N} a_j t_j x_j\right)^T x_n - \underbrace{\sum_{n=1}^{N} a_n t_n b}_{0}$$

$$= \sum_{n=1}^{N} a_n - \sum_{n=1}^{N} a_n t_n \left(\sum_{j=1}^{N} a_j t_j x_j\right)^T x_n$$

Putting it together

$$L = \frac{1}{2}\sum_{n=1}^{N} a_n^2 \|x_n\|^2 + \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{N} a_n t_n a_k t_k x_n^T x_k$$
$$+ \sum_{n=1}^{N} a_n - \sum_{n=1}^{N} a_n t_n \left(\sum_{j=1}^{N} a_j t_j x_j\right)^T x_n$$

$$= \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{N} a_n t_n a_k t_k x_n^T x_k$$

$$a_n \geq 0, \quad \sum_{n=1}^{N} a_n t_n = 0$$

New formulation of the Lagrangian

- New expression: $L = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{N} a_n t_n a_k t_k x_n^T x_k$

$a_n \geq 0, \quad \sum_{n=1}^{N} a_n t_n = 0$

- We want to maximize this with respect to a_n's.
- Note that the new form does not involve w!
- Ok, but it seems that the problem is more computationally difficult than before (N constraints for a_n's vs. d+1 constraints for b and for w of dimension d)

Advantage of the new formulation

- Consider the evaluation y(x) on a new point x:

$$y(x) = w^T x + b$$

$$y(x) = \left(\sum_{j=1}^{N} a_j t_j x_j\right)^T x + b$$

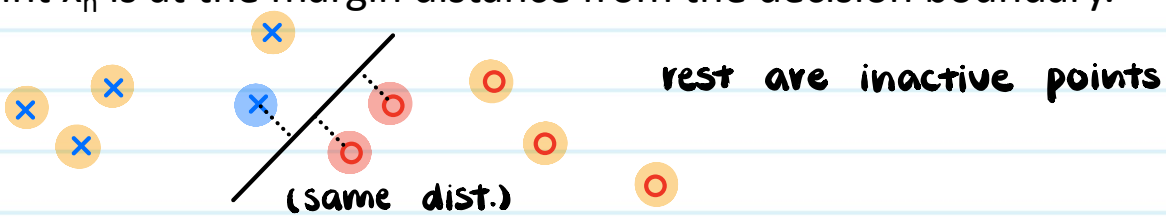
$$= \sum_{j=1}^{N} a_j t_j x_j^T x + b$$

- So we only need the non-zero a_n's! $(a_n \geq 0)$
- Recall active points ☺

$$a_n > 0$$

Recall active points

- Lagrange coefficient a_n is non-zero only if $t_n(w^T x_n + b) - 1 = 0$ i.e., only if the point x_n is at the margin distance from the decision boundary.
- Picture:



- Consequence: Most points will indeed have a_n zero, except for a few points that are on the margin distance from the decision boundary.
  - Subtle point: Only margin distance points can have a_n non-zero, but not all of them need to as long as there is at least one on each side with a_n non-zero.

Support vectors

- Points x_n that have a_n non-zero are called support vectors. Hence the name of the method.

- Now, recall the evaluation y(x) at test time:

$$y(x) = w^T x + b$$
$$= \sum_{n \in S} a_n t_n x_n^T x + b$$

S: set of support vectors

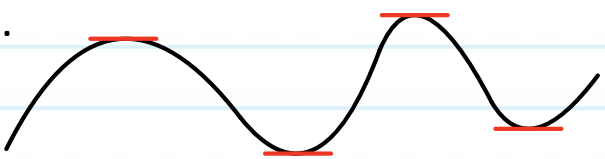
- Generally, very fast!

Transformation from primal to dual

- What we did in effect was the following:

$$\min_{w,b} \max_{a_n} L \rightarrow \max_{a_n} \min_{w,b} L$$

- Note that the solution for the dual is a lower bound for the solution of the primal.
- Picture:



- When the so-called KKT conditions are satisfied, the bound is tight.

KKT conditions

- At the saddle point solution, the following holds:
- Condition 1: $t_n(w^T x_n + b) \geq 1$ $\forall n$

  $a_n \geq 0$
- Condition 2: $\frac{\partial L}{\partial w} = 0$ $\frac{\partial L}{\partial b} = 0$ $\frac{\partial L}{\partial a_n} = 0$
- Condition 3: (complementary slackness)

  $a_n[1 - t_n(w^T x_n + b)] = 0$

  a_n is nonzero iff $1 - t_n(w^T x_n + b) = 0$

Back to our set-up

- Recall our vector w. $w = \sum_{n=1}^{N} a_n t_n x_n$
- Lagrange multipliers a_n's can be computed using standard methods such as sequential minimization optimization (SMO); forthcoming.
- Let's suppose that we have computed a_n's.
- What's left is to evaluate b.

Evaluation of the parameter b

- Directly: consider a SV x_n, $a_n > 0$

$$t_n(w^T x_n + b) = 1$$

- Numerically more stable:

$$\left(\sum_{n \in S} a_n t_n x_n\right)^T x_n + b = t_n$$

$$|S| \cdot b = \sum_{n \in S}\left(t_n - \left(\sum_{k \in S} a_k t_k x_k\right)^T\right) x_n$$

$$\Rightarrow b = \frac{1}{|S|}\left[\sum_{n \in S}\left(t_n - \left(\sum_{k \in S} a_k t_k x_k\right)^T\right) x_n\right]$$

Computing a_n's

- SMO is a standard technique. broader application beyond SVM
- Consider again the expression we want to maximize w.r.t. a_n's:

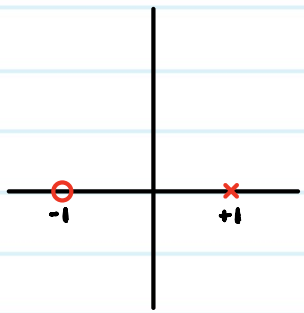
$$L = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{N} a_n t_n a_k t_k x_n^T x_k$$

$$a_n \geq 0 \qquad \sum_{n=1}^{N} a_n t_n = 0$$

- Initialize a_n's to some values so that constraints A and B hold.
- But setting all a_n's to zero does not work!

SMO on a small example

- Consider the following example with two data points, as follows:



$t_1^2 = 1, \quad t_2^2 = 1$

$$L = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{N} a_n t_n a_k t_k x_n^T x_k$$

$$= a_1 + a_2 - \frac{1}{2}(a_1^2 \|x_1\|^2 + a_2^2 \|x_2\|^2 - 2a_1 a_2 x_1^T x_2)$$

($\|x_1\|^2 \uparrow 1$, $\|x_2\|^2 \uparrow 1$, $x_1^T x_2 \underbrace{\quad}_{1}$)

from A: $a_1, a_2 \geq 0$

from B: $\sum_{n=1}^{N} a_n t_n = 0$

$a_1 - a_2 = 0$

$a_1 = a_2$

$$L = 2a_1 - \frac{1}{2}(a_1^2 + a_1^2 + 2a_1^2)$$

$$= 2a_1 - 2a_1^2$$

$$\frac{\partial L}{\partial a_1} = 0 \Rightarrow a_1 = \frac{1}{2}$$

$$\Rightarrow a_2 = \frac{1}{2}$$

SMO in general

- Pick two distinct points x_j and x_k with a_j and a_k
- Update from the current value by solving the quadratic.

isolate terms in L that have a_j / a_k

- Re-optimize as needed.

pick another 2 pts

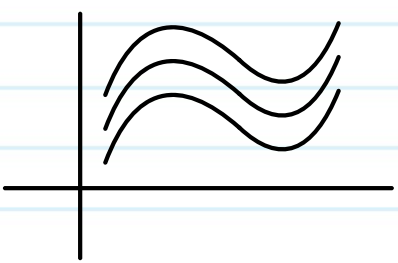
Beyond binary SVM classification

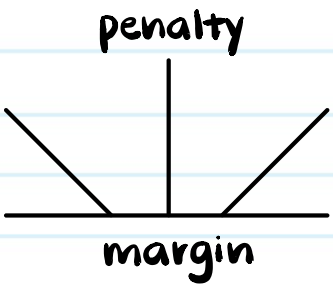
- For the multiclass classification:

  1 vs. 1

  1 vs. all

- SVM for regression





SVM summary

- Maximize the margin.
- Invariant to correctly classified points away from the margin (contrast with squared loss style minimization)
- What if the data is NOT linearly separable ?

ECE M146 Introduction to Machine Learning

Lecture 9 - Spring 2021

Prof. Lara Dolecek

ECE Department, UCLA

Today's Lecture

Recap:

- Support Vector Machines (SVM)

New topic:

- Kernels – theory
- Soft SVM
- Kernels – applications to familiar methods

Today's Lecture

Recap:

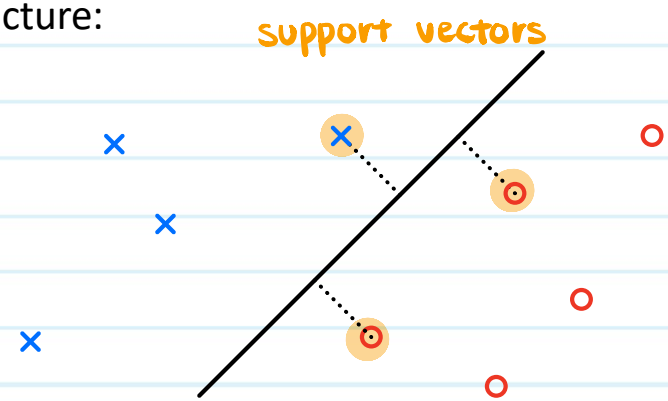
- Support Vector Machines (SVM)

New topic:

- Kernels – theory
- Soft SVM
- Kernels – applications to familiar methods

Recap: SVM

- Last time we introduced the SVM method for binary classification.
- This method finds the vector w so that the smallest distance from a training point to the decision boundary is maximized.
- Picture:



Recap: SVM

- We then formulated the minimization problem as a quadratic program with linear constraints.
- A way to solve such a problem is by converting a constrained optimization problem into an unconstrained optimization problem.
- This is done with a Lagrangian, which incorporates constraints with Lagrange multipliers.

Recap: SVM

- Note that we are here solving a minimization problem.
- Utility of the Lagrangian extends to maximization problems as well as minimization/maximization problems with inequality and/or equality constraints.
  - The difference is in the sign.

Recap: SVM

- Even though the we could have solved the unconstrained problem as stated (primal), we intentionally converted it into a different problem (dual).
- Math: $$L = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{N} a_n t_n a_k t_k x_n^T x_k$$
- The advantage of the dual formulation is that all the terms are scaled versions of $x_n^T x_k$.

Recap: SVM

- At test time

$$w = \sum_{n \in S} a_n t_n x_n$$

S: set of indices corresponding to support vectors

$|S| \ll N$

given test pt x,

$w^T x + b$

compute $\sum_{n \in S} a_n t_n x_n^T x + b$

Today's Lecture

Recap:

- Support Vector Machines (SVM)

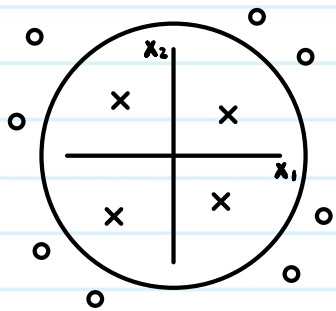
New topic:

- Kernels – theory
- Soft SVM
- Kernels – applications to familiar methods

Achieving linear separability in higher dimensions

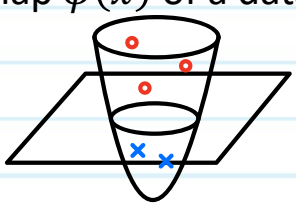
- Consider the following example:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$



- If we use the original 2-d vector representation, the data is clearly not linearly separable.
- Now, consider, the following map $\varphi(x)$ of a data point x:

$$\varphi(x) = \begin{bmatrix} x_1 \\ x_2 \\ x_1 + x_2 \end{bmatrix}$$



Achieving linear separability in higher dimensions

- By adding new features, we made the classes linearly separable.
- But does this only mean that we need to do more computations ?
- Let's see.

Feature expansion example

- Let's consider the following example. Suppose x and z are given as:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

- Now suppose that their maps are as follows:

$$\varphi(x) = \begin{bmatrix} 1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_1x_2, x_2x_1, x_2^2 \end{bmatrix}^T$$

$$\varphi(z) = \begin{bmatrix} 1, \sqrt{2}z_1, \sqrt{2}z_2, z_1^2, z_1z_2, z_2z_1, z_2^2 \end{bmatrix}^T$$

- We went from having 2 to having 7 dimensions!

Example, continued

- Compute the inner product $\varphi(x)^{\mathsf{T}}\varphi(z)$:

$$\varphi(x)^T\varphi(z) = 1 + 2x_1z_1 + 2x_2z_2 + x_1^2z_1^2 + 2x_1x_2z_1z_2 + x_2^2z_2^2$$

- Compare to $(1 + x^T z)^2$:

$$1 + 2x^Tz + (x^Tz)^2$$
$$= 1 + 2x_1z_1 + 2x_2z_2 + (x_1z_1 + x_2z_2)^2$$
$$= 1 + 2x_1z_1 + 2x_2z_2 + x_1^2z_1^2 + 2x_1x_2z_1z_2 + x_2^2z_2^2$$

- Same answer, but the **latter** is much faster. Kernel trick!

(x^Tz) + optimizations on scalars

Kernels

for $d=2$, $\varphi(x)$ has dim = 7

- There are many valid kernels, beyond example we just saw.
- Polynomial kernel:

$$k(x,z) = (1 + x^T z)^d$$

  - Previous example was for d=2.
- Gaussian kernel: radial basis function

$$k(x,z) = e^{-\gamma \|x-z\|^2}$$

$\gamma > 0$

recall: $e^x = \sum \frac{x^n}{n!}$

When does the map $\varphi(x)$ produce a valid kernel ?

- Suppose we have N data points.
- Let K_{ij} be the following: $\varphi(x_i)^T \cdot \varphi(x_j) = K_{ij}$

  ex. $K_{ij} = (1 + x_i^T x_j)^d$

- Note that with this representation we do not need to specify the map explicitly!
- The overall matrix K is NxN and symmetric.

$$\varphi(x_i)^T \cdot \varphi(x_j) = \varphi(x_j)^T \cdot \varphi(x_i)$$

Properties of this matrix

$$y = [y_1, y_2, \cdots, y_N]^T$$

- Consider the following:

$$y^T K y = [y_1\ y_2 \cdots y_N] \begin{bmatrix} K_{11} & K_{12} & \cdots & K_{1N} \\ K_{21} & K_{22} & & K_{2N} \\ \vdots & \vdots & \cdots & \vdots \\ K_{N1} & K_{N2} & \cdots & K_{NN} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

$$= \left[\sum_{j=1}^{N} y_j k_{j1} \quad \sum_{j=1}^{N} y_j k_{j2} \quad \cdots \quad \sum_{j=1}^{N} y_j k_{jN}\right] \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{N} y_i \varphi(x_i)^T \varphi(x_j) x_j$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{N} y_i \left[\sum_{k=1}^{|\varphi|} \varphi_k(x_i)\varphi_k(x_j)\right] y_j \geq 0$$

- Therefore, the matrix is positive semi-definite (PSD).

$\varphi_k(x_i)$: k^{th} component of $\varphi(x_i)$

Properties of this matrix

K:

- square $(k \times k)$
- symmetric
- $\forall\ y,\ y^T K y \geq 0$

↑ K is PSD

Example

- Let's consider the Gaussian kernel again, for 2-d vectors:

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

$$\gamma > 0$$

$$K = \begin{bmatrix} 1 & t \\ t & 1 \end{bmatrix}$$

$$K_{11} = K(x_1, x_1) = \exp(0) = 1$$

- Show PSD property:

$$y^T K y = \begin{bmatrix} y_1 & y_2 \end{bmatrix} \begin{bmatrix} 1 & t \\ t & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$= \begin{bmatrix} y_1 + y_2 t & y_1 t + y_2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$= y_1^2 + 2 y_1 y_2 t + y_2^2$$

$$= A(y_1^2 + 2 y_1 y_2 + y_2^2) + (1 - t)(y_1^2 + y_2^2)$$

$$= \underbrace{A(y_1 + y_2)^2}_{\geq 0} + \underbrace{(1 - t)}_{>0}\underbrace{(y_1^2 + y_2^2)}_{>0}$$

$$> 0$$

If $y^T A y > 0 \quad \forall \; y \neq 0$, then A is PD.

Another example

- Now consider the following example

is $\|x-z\|^2$ valid kernel?

let's consider 2D vector and $K = \begin{bmatrix} 0 & t \\ t & 0 \end{bmatrix}$

$$y^T K y = \begin{bmatrix} y_1 & y_2 \end{bmatrix} \begin{bmatrix} 0 & t \\ t & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$= \begin{bmatrix} y_2 t & y_1 t \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$= 2 y_1 y_2 t$$

if $y = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$, then $y^T K y < 0$

$\Rightarrow K$ is not PSD

More on kernels

PSD property of the matrix k:

1. if k is valid kernel, then it is PSD

2. any PSD matrix can be associated w/ a feature map $f(x)$

3. if K_1 and K_2 are valid kernels, so is $K_1 + K_2$ and $K_1 \cdot K_2$

Kernels in SVM

$$L = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{N} a_n t_n a_k t_k x_n^T x_k$$

$$w = \sum_{n \in S} a_n t_n x_n$$

at test time

$$w^T x + b \gtrless 0 \quad \begin{matrix} \text{class } +1 \\ \text{class } -1 \end{matrix}$$

$$\sum_{n \in S} a_n t_n \underbrace{x_n^T \cdot x}_{\varphi(x_n)^T \varphi(x)} + b \gtrless 0$$

$$K(x_n, x)$$

Today's Lecture

Recap:

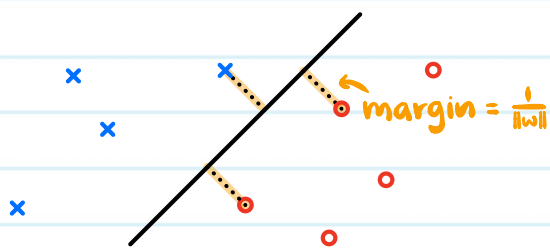
- Support Vector Machines (SVM)

New topic:

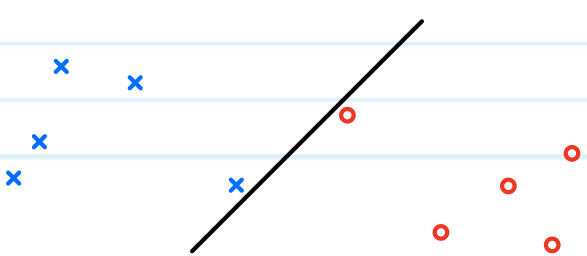
- Kernels – theory
- Soft SVM
- Kernels – applications to familiar methods

Soft margin SVM

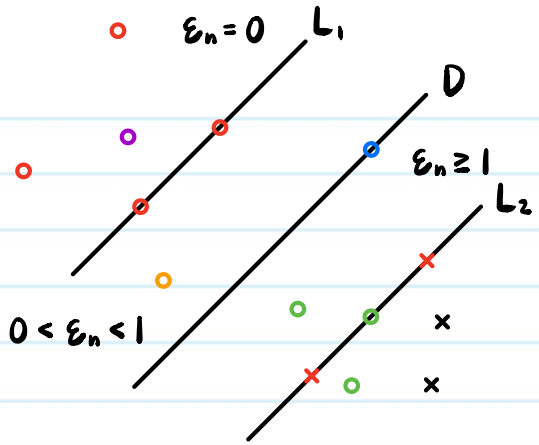
- Recall SVM for linearly separable data:



- But if the data is (almost) not linearly separable, we are forcing a decision boundary that is not good – poor generalization error.



Instead, let's allow for slack

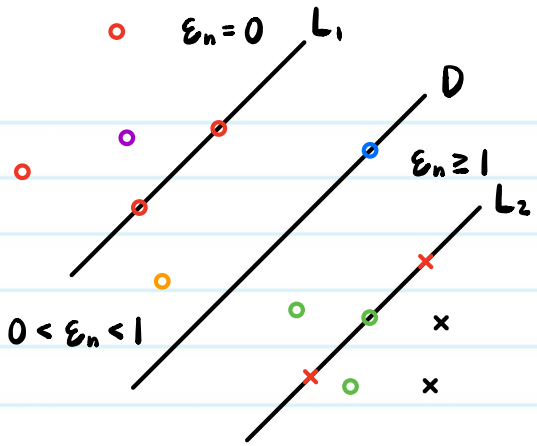


- New linear constraints:

$$t_n(w^T \cdot x_n + b) \geq 1 - \xi_n \quad , \quad \xi_n \geq 0$$

- Case 1: same as hard SVM: if x_n is on L_1/L_2 and correctly classified, $t_n(w^T \cdot x_n + b) = 1 \Leftrightarrow \xi_n = 0$
- Case 2: if x_n is on D: $y(x_n) = 0$
  $t_n(w^T \cdot x_n + b) = 0 \Leftrightarrow \xi_n = 1$
- Case 3: if $0 < t_n \cdot y(x_n) < 1$: $0 < \xi_n < 1$

Instead, let's allow for slack



- New linear constraints:

$$t_n(w^T \cdot x_n + b) \geq 1 - \xi_n \quad , \quad \xi_n \geq 0$$

- Case 4: (x_n, t_n) incorrectly classified:

$$t_n \cdot y(x_n) < 0 \iff \xi_n > 1$$

- Case 5: (x_n, t_n) correctly classified:

$$t_n \cdot y(x_n) > 1 \iff \xi_n = 0$$

Soft margin SVM

- Formulation:

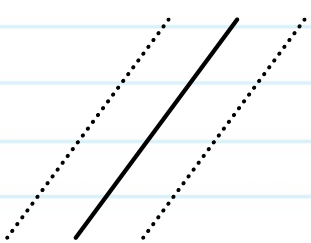
$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}\varepsilon_n, \qquad t_n(w^T \cdot x_n + b) \geq 1 - \varepsilon_n, \quad \varepsilon_n \geq 0$$
$$C > 0$$

Role of the hyperparameter C

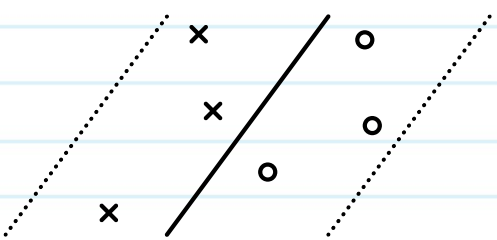
if $C = \infty \Rightarrow$ we recover the original SVM formula

$\Rightarrow \xi_n = 0$

large C : narrow margin



small C : wide margin



Support vectors in soft SVM

- Now, the support vectors will be all the points on the margin and on the “penalty” side of it (we’ll see this shortly).
- There are typically more support vectors in soft SVM than in the original (hard) SVM.

Reformulation of the Lagrangian

- Before we had:

$$L = \frac{1}{2}\|w\|^2 + \sum_{n=1}^{N} a_n(1 - b_n(w^T x_n + b))$$

- Now we have:

$$L = \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}\varepsilon_n + \sum_{n=1}^{N} a_n(1 - \varepsilon_n - t_n(w^T x_n + b)) - \sum_{n=1}^{N} b_n\varepsilon_n$$

max L: $b_n = \infty$

want $\varepsilon_n \geq 0$

(otherwise, violation)

  - Lagrange multipliers:

$a_n \geq 0$ (from before)

$b_n \geq 0$ (new)

to ensure $\varepsilon_n \geq 0$

KKT conditions

- Partial derivatives
- Old conditions are still here:

$$\frac{\partial L}{\partial w} = 0 \quad \Rightarrow \quad w = \sum_{n \in S} a_n t_n x_n$$

$$\frac{\partial L}{\partial b} = 0 \quad \Rightarrow \quad \sum_{n=1}^{N} a_n b_n = 0$$

$$\frac{\partial L}{\partial a_n} = 0$$

- New conditions are:

$$\frac{\partial L}{\partial \xi_n} = C - a_n - b_n = 0$$

KKT conditions

- Complementary slackness, revised:

$$a_n(t_n(w^T x_n + b) - 1 + \varepsilon_n) = 0$$

- These are new:

$$b_n \varepsilon_n = 0$$

- Inequalities, old and new:

$$a_n \geq 0 \qquad b_n \geq 0$$

$$t_n(w^T \cdot x_n + b) \geq 1 - \varepsilon_n$$

Expression for the Lagrangian

- We can eliminate w, b, ε_n from the expression (similar to the case we had before) to obtain:

$$L = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{N} a_n t_n a_k t_k x_n^T x_k$$

$$a_n \geq 0 \qquad b_n \geq 0$$

$$\sum_{n=1}^{N} a_n b_n = 0$$

- Compact representation:

$$a_n + b_n = c$$

$$a_n \leq c \Rightarrow 0 \leq a_n \leq c$$

Support vectors

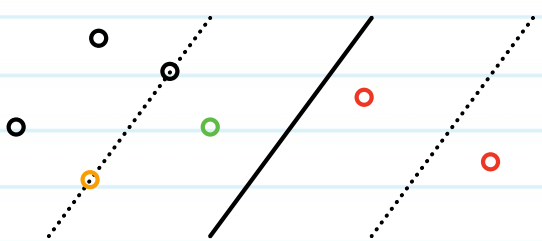
- If $a_n = 0$ $b_n = c \Rightarrow \varepsilon_n = 0$

- If $a_n \neq 0$ $a_n > 0$

$$a_n(t_n(w^T x_n + b) - 1 + \varepsilon_n) = 0$$

$$\Rightarrow \quad t_n(w^T x_n + b) - 1 + \varepsilon_n = 0$$

①	②
$0 < a_n < c$	$a_n = c$
$0 < b_n < c$	$b_n = 0$
$\varepsilon_n = 0$	$0 \leq \varepsilon_n \leq 1$ $\varepsilon_n > 1$



Today's Lecture

Recap:

- Support Vector Machines (SVM)

New topic:

- Kernels – theory
- Soft SVM
- Kernels – applications to familiar methods

Kernels in regression

- Recall the regression set-up

Kernalized Linear Regression

When can kernal trick be used:

① Model parameter only depends on inner products of features.

② At test time, the result also only depends on inner product.

$$w = (X^T X + \lambda I)^{-1} X^T y \qquad \text{X Kernal trick}$$

Rewriting

$$(X^T X + \lambda I)\, w = X^T y$$

$$w = \frac{1}{\lambda}(X^T y - X^T X w) = X^T \cdot \underbrace{\frac{1}{\lambda}(y - Xw)}_{a}$$

$$w = X^T a$$

$$\lambda a = Y - X X^T a$$

$$a = (X X^T + \lambda I)^{-1} y$$

$X X^T \to \begin{bmatrix} x_1^T x_1 & \cdots \\ x_1^T x_2 & \\ \vdots & \ddots \end{bmatrix} \to \begin{bmatrix} k(x_i, x_j) & \cdots \\ \vdots & \ddots \end{bmatrix}$

At test time:

$$w^T z = X^T a \cdot z = a^T X^T z$$

$$= \sum_{i=1}^{N} a_i \underbrace{x_i^T z}_{k(x_i, z)}$$

Kernels in perceptron

$w^{(i+1)} = w^{(i)} + x_n y_n$ (when misclassified)

$w = a_1 x_1 y_1 + a_2 x_2 y_2 + \ldots$

$= \sum_i a_i x_i y_i$

at test time:

$\text{sign}(w^T z) = \text{sign}(\sum_i a_i \underbrace{x_i^T z}_{K(x_i, z)} y_i)$

Summary of the recent parametric methods based on discriminative modeling

- Perceptron
- Logistic Regression
- Linear Least Squares
- SVM

- Next: probabilistic generative modeling