CS584 – Machine Learning

Topic: Support Vector Machines

Mustafa Bilgic

http://www.cs.iit.edu/~mbilgic

https://twitter.com/bilgicm

References

- Great notes, by Andrew Ng
  - http://cs229.stanford.edu/notes/cs229-notes3.pdf
- Great lecture, by Patrick Winston
  - https://www.youtube.com/watch?v=_PwhiWxHK8o

Objective Function

- $D = \{\langle x^{(d)}, y^{(d)} \rangle\}$ where $y \in \{-1, +1\}$
- Find w and b such that
  - $w^T x^{(d)} + b \geq +1$ if $y^{(d)} = +1$ and
  - $w^T x^{(d)} + b \leq -1$ if $y^{(d)} = -1$
- Which can be written simply
  - $y^{(d)}\left(w^T x^{(d)} + b\right) \geq +1$

Functional Margin

- The functional margin of a single instance is
  - $\hat{\gamma}^{(d)} = y^{(d)}\left(w^T x^{(d)} + b\right)$
- The minimum functional margin with respect to $D = \left\{\left\langle x^{(d)}, y^{(d)}\right\rangle\right\}$ is
  - $\hat{\gamma} = \min_{d \in D} \hat{\gamma}^{(d)}$
- We can try to maximize $\hat{\gamma}$ so that all instances are at least as far away as $\hat{\gamma}$
- One problem
  - Rescaling w and b by a constant a also rescales $\hat{\gamma}$

Geometrical Margin

- What is the geometrical distance to the margin?
- First, prove that w is perpendicular to the separating hyperplane
  - See OneNote
- Then, prove that $x^{(d)}$'s distance to the hyperplane is
  - $\gamma^{(d)} = \frac{|w^T x^{(d)} + b|}{\|w\|}$, which can also be written as
  - $\gamma^{(d)} = \frac{|w^T x^{(d)} + b|}{\|w\|} = \frac{y^{(d)}(w^T x^{(d)} + b)}{\|w\|} = y^{(d)}\left(\left(\frac{w}{\|w\|}\right)^T x^{(d)} + \frac{b}{\|w\|}\right)$
  - See OneNote
  - Note that when $\|w\| = 1$, functional margin is equal to the geometric margin
  - Furthermore, note that geometric margin is invariant to rescaling w and b

Geometrical Margin

- The minimum geometrical margin with respect to $D = \{\langle x^{(d)}, y^{(d)} \rangle\}$ is
  - $\gamma = \min_{d \in D} \gamma^{(d)}$
- We can write our objective function as follows
  - $\max_{\gamma, w, b} \gamma$
  - subject to
  - $y^{(d)}(w^T x^{(d)} + b) \geq \gamma$ for $d \in D$
  - $\|w\| = 1$
- $\|w\| = 1$ constraints ensures that functional and geometrical margin are equal and hence maximizing geometrical margin is equivalent to maximizing functional margin
- Problem: $\|w\| = 1$ is a non-convex constraint

How About?

- $\max_{\hat{\gamma},w,b} \frac{\hat{\gamma}}{\|w\|}$
- subject to
- $y^{(d)}\left(w^T x^{(d)} + b\right) \geq \hat{\gamma}$ for $d \in D$
- Remember that geometric margin γ and functional margin $\hat{\gamma}$ are related as $\gamma = \frac{\hat{\gamma}}{\|w\|}$ and hence this objective function is still maximizing the geometrical margin and does not have the non-convex constraint $\|w\| = 1$
- Problem: this time, the objective function is non-convex

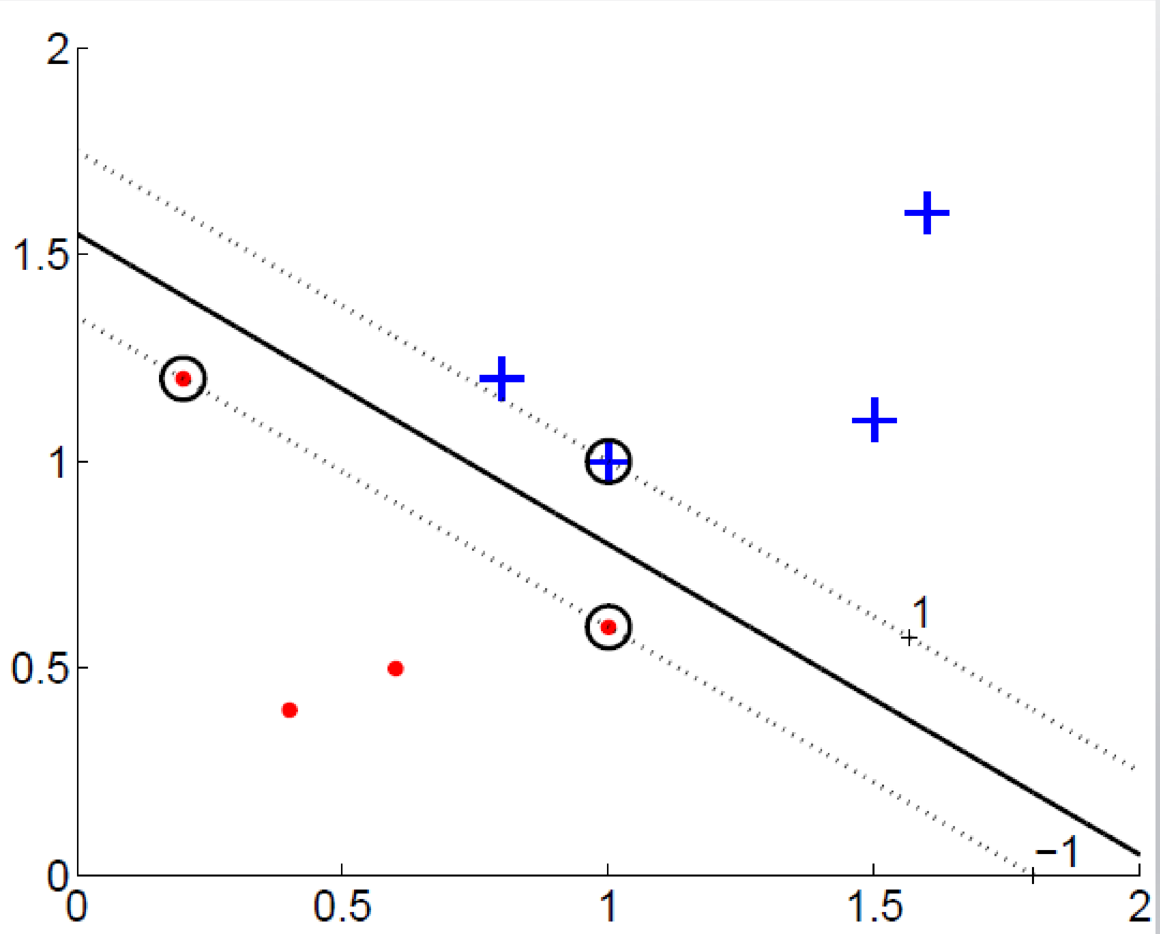
FINALLY ☺

- Remember that the geometric margin is invariant to rescaling w and b, and thus, we can scale them any way we want without effecting the geometric margin
  - Thus, we put the constraint that the minimum functional margin of w and b with respect to D is 1. That is, $\hat{\gamma} = 1$. Then, our objective function becomes
- $\max_{w,b} \frac{1}{\|w\|}$
- subject to
- $y^{(d)}\left(w^T x^{(d)} + b\right) \geq 1$ for $d \in D$
- This is convex quadratic objective function with linear constraints and quadratic programming solvers can easily solve it. QED.
- For example: http://cvxopt.org/userguide/coneprog.html#quadratic-programming

Maximum Margin Classification

- $\min \frac{1}{2} w^T w$ subject to $y^{(d)}\left(w^T x^{(d)} + b\right) \geq +1$

Margin



Credit: Ethem Alpaydin. Introduction to Machine Learning. 3rd Edition. *http://www.cmpe.boun.edu.tr/~ethem/i2ml3e*

Lagrangian - Primal

- The objective function
  - $\min \frac{1}{2} w^T w$ subject to $y^{(d)}\left(w^T x^{(d)} + b\right) \geq +1$
- Form its Lagrangian
  - $L_p = \frac{1}{2} w^2 - \sum_{d \in D} \alpha^{(d)}\left(y^{(d)}\left(w^T x^{(d)} + b\right) - 1\right)$
  - $\alpha^{(d)} \geq 0 \; \forall d \in D$
- Take its derivative wrt w and b
  - $\frac{\partial L_p}{\partial w} = w - \sum_{d \in D} \alpha^{(d)} y^{(d)} x^{(d)} = 0 \Rightarrow w = \sum_{d \in D} \alpha^{(d)} y^{(d)} x^{(d)}$
  - $\frac{\partial L_p}{\partial b} = \sum_{d \in D} \alpha^{(d)} y^{(d)} = 0$
- See OneNote

LAGRANGIAN - DUAL

- Enforce the derivatives in the primal itself
- $L_{dual} =$
  $-\frac{1}{2}\sum_{i\in D}\sum_{j\in D}\alpha^{(i)}\alpha^{(j)}y^{(i)}y^{(j)}\,x^{(i)}x^{(j)} + \sum_{d\in D}\alpha^{(d)}$
- Subject to
  - $\sum_{d\in D}\alpha^{(d)}\,y^{(d)} = 0$
  - $\alpha^{(d)} \geq 0\ \forall d \in D$
- See OneNote
- Maximize L_{dual} with respect to α
- Use cvxopt.solvers.QP (next slide)

CVXOPT.SOLVERS.QP

- http://cvxopt.org/userguide/coneprog.html#quadratic-programming
- Cvxopt solves
  - minimize $\frac{1}{2}x^T Px + q^T x$
  - subject to
  - $G(x) \leq h$
  - $A(x) = b$
- Let's re-write cvxopt's formalism in terms of α
  - minimize $\frac{1}{2}\alpha^T P\alpha + q^T \alpha$
  - subject to
  - $G(\alpha) \leq h$
  - $A(\alpha) = b$
- Let's look at the correspondence between L_{dual} and this formalism

DUAL IN CVXOPT

- Cvxopt
  - minimize $\frac{1}{2}\alpha^T P\alpha + q^T\alpha$
  - subject to
  - $G(\alpha) \leq h$
  - $A(\alpha) = b$
- Maximize dual ≡ minimize minus of dual
  - $-L_{dual} = \frac{1}{2}\sum_{i\in D}\sum_{j\in D}\alpha^{(i)}\alpha^{(j)}y^{(i)}y^{(j)}x^{(i)}x^{(j)} - \sum_{d\in D}\alpha^{(d)}$
  - Subject to
    - $\alpha^{(d)} \geq 0\ \forall d \in D$ which we will rewrite as $-\alpha^{(d)} \leq 0\ \forall d \in D$
    - $\sum_{d\in D}\alpha^{(d)}y^{(d)} = 0$
- Correspondence
  - $\frac{1}{2}\sum_{i\in D}\sum_{j\in D}\alpha^{(i)}\alpha^{(j)}y^{(i)}y^{(j)}x^{(i)}x^{(j)}$ is $\frac{1}{2}\alpha^T P\alpha$
  - $-\sum_{d\in D}\alpha^{(d)}$ is $q^T\alpha$
  - $-\alpha^{(d)} \leq 0\ \forall d \in D$ is $G(\alpha) \leq h$
  - $\sum_{d\in D}\alpha^{(d)}y^{(d)} = 0$ is $A(\alpha) = b$

Python Code – No Kernels – Linearly Separable

```
import numpy as np
import cvxopt
X = np.array([[1, 1], [1, 3], [3, 1], [3, 3]])
y = np.array([1., 1., -1., -1.])
ni, nf = X.shape
X_ = np.matrix(X)
P=cvxopt.matrix(np.outer(y,y)*np.array((X_*X_.T)))
q=cvxopt.matrix(-1*np.ones(ni))
G=cvxopt.matrix(np.diag(-1*np.ones(ni)))
h=cvxopt.matrix(np.zeros(ni))
A=cvxopt.matrix(y, (1, ni))
b=cvxopt.matrix(0.0)
sol=cvxopt.solvers.qp(P, q, G, h, A, b)
alphas = np.ravel(sol['x'])
```

Primal VS Dual

- Primal and dual formulation lead to the same solution under certain conditions, called
  - Karush–Kuhn–Tucker conditions
  - Often abbreviated as KKT conditions
  - We will not go into details of KKT in this class
- Dual has the nice formalism that it enables the "kernel" trick
  - More on this soon
  - First, see Jupyter Notebook for a solution to the dual formalism

Support Vectors

- $\alpha^{(d)}$ is non-zero for instances that have a functional margin of $+1$ or -1 and zero for others
- The ones that have a functional margin of $+1$ or -1 are called the support vectors
- b can be calculated using support vectors
- For classification, all we need is the support vectors and their α values

Kernel Trick

- See OneNote for explanation
- See Jupyter Notebook for examples

Some Kernels

- Linear kernel
  - $K\left(x^{(i)}, x^{(j)}\right) = {x^{(i)}}^T x^{(j)}$
- Polynomial kernel degree d
  - $K\left(x^{(i)}, x^{(j)}\right) = \left({x^{(i)}}^T x^{(j)} + 1\right)^d$
- Gaussian kernel
  - $K\left(x^{(i)}, x^{(j)}\right) = e^{-\frac{\left\|x^{(i)} - x^{(j)}\right\|^2}{2s^2}}$

Soft Margin

- What if the data is not linearly separable?
  - One solution is obviously to use non-linear kernels
- What if the data is not separable even with a kernel?
- Or, what if, we do not want to overfit
- A solution is to relax the hard constraints a bit
- New objective function
  - $\min \frac{1}{2} w^T w + C \sum_{d \in D} \xi^{(d)}$
  - subject to
  - $y^{(d)}\left(w^T x^{(d)} + b\right) \geq +1 - \xi^{(d)}$
- Formulate Lagrangian Primal and Dual
  - See OneNote
- Examples
  - See Jupyter Notebook

Scikit-Learn

- http://scikit-learn.org/stable/modules/svm.html