PRINCETON UNIVERSITY
ORFE
Instructor:
Amir Ali Ahmadi

This lecture:

- Operations that preserve convexity
  - Nonnegative weighted sum
  - Pointwise maximum
  - Composition with an affine mapping
  - Restriction to a line
- Applications of convex optimization to statistics and machine learning
  - Least absolute shrinkage and selection operator (LASSO); aka least squares with l_1 penalty
  - Supervised learning
    - Support vector machines (SVMs)

Operations that preserve convexity

- In the previous lecture, we covered some of the reasons why convex optimization problems are so desirable in the field of optimization. We also gave some characterizations of convex functions that made it easier to recognize convex problems.
- Nevertheless, since testing convexity can in general be an intractable task [AOPT13], it is useful to produce as many convex functions as we can from a ground set of functions that we already know are convex.
- This is exactly what "convexity-preserving rules" do: They take some convex functions as input and perform certain operations to produce more convex function. Often, the new convex functions turn out to have a much richer class of applications.
- There is a long list of convexity-preserving rules [BV04]. We present only four of them here. The software CVX that you are using has a lot of these rules built in [BG08], [CVX11].

Rule 1: Nonnegative weighted sums

If $f_1, \dots, f_n$ are convex functions and $\omega_1, \dots, \omega_n \geq 0$, then

$$f(x) = \omega_1 f_1(x) + \cdots + \omega_n f_n(x)$$

is convex also. Similarly, a nonnegative weighted sum of concave functions is concave.

Proof: Let $f_1, \dots, f_n$ be convex functions, $\omega_1, \dots, \omega_n \geq 0$, $x, y \in \mathbb{R}^n$, and $\lambda \in [0,1]$. Then:

$$f(\lambda x + (1-\lambda)y) = \omega_1 f_1(\lambda x + (1-\lambda)y) + \cdots + \omega_n f_n(\lambda x + (1-\lambda)y)$$
$$\leq \omega_1 \cdot \big(\lambda f_1(x) + (1-\lambda) f_1(y)\big) + \cdots + \omega_n \cdot \big(\lambda f_n(x) + (1-\lambda) f_n(y)\big)$$
$$= \lambda \big(\omega_1 f_1(x) + \cdots + \omega_n f_n(x)\big) + (1-\lambda) \cdot \big(\omega_1 f_1(y) + \cdots + \omega_n f_n(y)\big)$$
$$= \lambda f(x) + (1-\lambda) f(y),$$

where the second line is obtained using convexity of $f_1, \dots, f_n$ and the fact that the inequalities are preserved as $\omega_1, \dots, \omega_n$ are nonnegative. □

- Note that this in particular implies:
  - If $\alpha \geq 0$ and f is convex, then αf is convex.
  - If f_1 and f_2 are convex, then $f_1 + f_2$ is convex.
- Also easy to prove the theorem from the second order characterization of convexity (assuming differentiability). Do you see how the proof would work?
- Since the sum of two convex functions is convex, a constraint of the following form is a valid CVX constraint (why?):
  - Convex function $\leq$ Concave function.

- **Q:** If f_1, f_2 are convex functions,
  - is $f_1 - f_2$ convex? $f_1 = 0,\ f_2 = x^2$
  - is $f_1 \times f_2$ convex? $f_1 = x,\ f_2 = x^2$
  - is $\frac{f_1}{f_2}$ convex? $f_1 = x^{3/2},\ f_2 = x$

Rule 2: Composition with an affine mapping

Suppose $f: \mathbb{R}^n \to \mathbb{R}$, $A \in \mathbb{R}^{n \times m}$ and $b \in \mathbb{R}^n$. Define $g: \mathbb{R}^m \to \mathbb{R}$ by

$$g(x) = f(Ax + b)$$

with $dom(g) = \{x | Ax + b \in dom(f)\}$. Then, if f is convex, so is g; if f is concave, so is g.

The proof is given on the following page.

Proof: Let $x, y \in \mathbb{R}^n$ and $\lambda \in [0,1]$. Then:

$$\begin{aligned} g(\lambda x + (1-\lambda)y) &= f(A(\lambda x + (1-\lambda)y) + b) \\ &= f(\lambda \cdot Ax + (1-\lambda) \cdot Ay + \lambda b + (1-\lambda)b) \\ &= f\big(\lambda \cdot (Ax+b) + (1-\lambda) \cdot (Ay+b)\big) \\ &\leq \lambda f(Ax+b) + (1-\lambda) f(Ay+b) \text{ using the fact that } f \text{ is convex} \\ &= \lambda g(x) + (1-\lambda) g(y). \end{aligned}$$

So g is convex. The proof in the concave case is similar.

Example.
The following function is immediately seen to be convex. (Without knowing the rule above, it would be much harder to prove convexity.)

$$f(x_1, x_2) = (x_1 - 2x_2)^4 + 2e^{(3x_1 + 2x_2 - 5)}.$$

z^4 (convex), composed with $x_1 - 2x_2$ (affine)

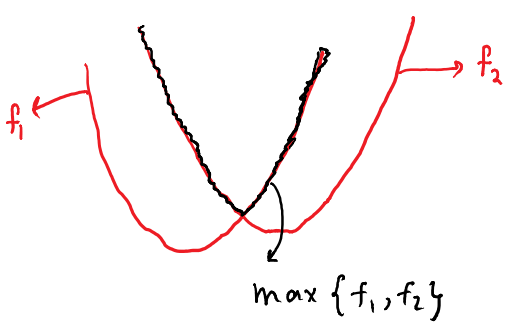
e^z (convex), composed with $3x_1 + 2x_2 - 5$ (affine)

Rule 3: Pointwise maximum

If $f_1, \ldots, f_m$ are convex functions then their pointwise maximum

$f(x) = \max\{f_1(x), f_2(x), \ldots f_m(x)\}$,

with $dom(f) = dom(f_1) \cap dom(f_2) \cap \cdots \cap dom(f_m)$, is also convex.



Proof: Pick any $x, y \in \text{dom}(f)$, $\lambda \in [0,1]$. Then,

$$f(\lambda x + (1-\lambda)y) = f_j(\lambda x + (1-\lambda)y) \quad \text{(for some } j \in \{1,\dots,m\}\text{)}$$

f_j convex $\Rightarrow \lambda f_j(x) + (1-\lambda) f_j(y)$

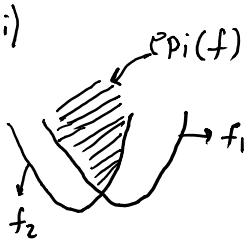
$$\leq \lambda \max\{f_1(x),\dots,f_m(x)\} + (1-\lambda)\max\{f_1(x),\dots,f_m(x)\}$$

$$= \lambda f(x) + (1-\lambda) f(y). \quad \square$$

Also easy to prove from epigraphs.

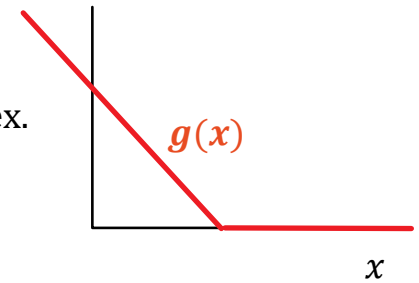
Recall f convex $\Leftrightarrow$ epi(f) convex. $\text{epi}(f) = \bigcap_{i=1}^{m} \text{epi}(f_i)$

And we know intersection of convex sets is convex. $\square$



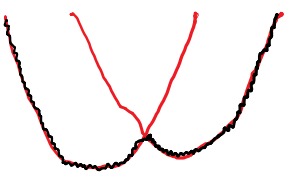
Example - the hinge loss

The hinge loss function $g(x) = \max(0, 1-x)$ is convex.



- One can similarly show that the pointwise minimum of two concave functions is concave.

- But the pointwise minimum of two convex functions may not be convex.



Rule 4: Restriction to a line

Let $f: \mathbb{R}^n \to \mathbb{R}$ be a convex function and fix some $x, y, \in \mathbb{R}^n$. Then, the function $g: \mathbb{R} \to \mathbb{R}$ given by $g(\alpha) = f(x + \alpha y)$ is convex.

Proof: This directly follows from Rule 2 (composition with an affine mapping). Indeed, $x + ty$ is an affine expression in t since x and y are fixed. Here is a second independent proof:

Suppose for some x, y, $g(\alpha) = f(x+\alpha y)$ was not convex.

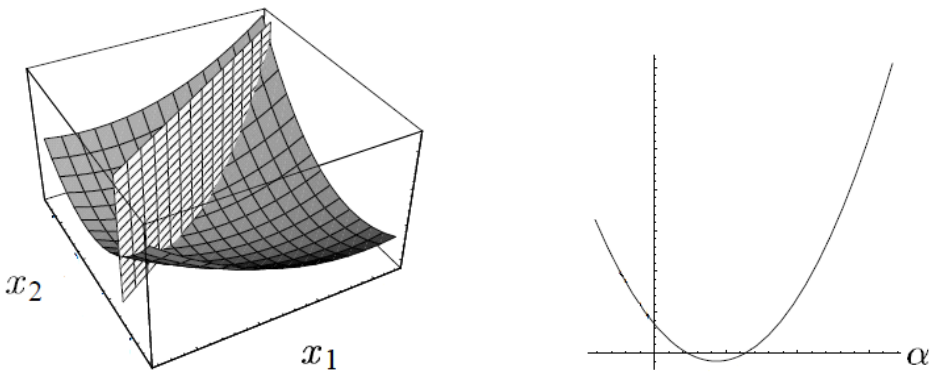
$\Rightarrow \exists \lambda \in [0,1], \alpha_1, \alpha_2$ s.t. $g(\lambda\alpha_1 + (1-\lambda)\alpha_2) > \lambda g(\alpha_1) + (1-\lambda) g(\alpha_2)$.

$\Rightarrow f(x + (\lambda\alpha_1 + (1-\lambda)\alpha_2)y) = f(\lambda(x+\alpha_1 y) + (1-\lambda)(x+\alpha_2 y)) > \lambda f(x+\alpha_1 y) + (1-\lambda) f(x+\alpha_2 y)$.

□

Restriction to a line geometrically:

Image credit: [She94]



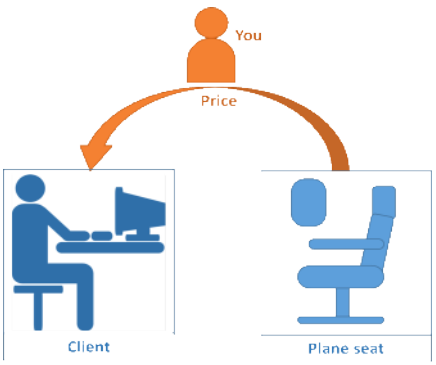
$f(x_1, x_2)$ $\qquad$ $f(\alpha)$

- Many of the algorithms we will see in future lectures work by iteratively minimizing a function over lines. It's useful to remember that the restriction of a convex function to a line remains convex. Hence, in each subproblem where we are faced with a univariate minimization problem, we can just set the derivative equal to zero and be sure that that produces a global minimum.

Some applications of convex optimization in statistics and machine learning

Personalized pricing of flight tickets

You have been hired as a quantitative analyst by Priceline.com, a major travel website company. A distinguishing feature of the company is its "Name Your Own Price" tool, a mechanism for customers to bid on the price they want to pay for a flight ticket. If the bid is high enough, the ticket is sold. If it's too low, the company displays a rejection message with a counteroffer. You have been hired to optimize this pricing strategy. (After all, an A on a course called "Computing and Optimization" from Princeton makes your boss think you can optimize anything!) Faced with this task, you want an efficient mechanism for predicting the highest price a given customer is willing to pay for a flight.



As usual in the age of big data and no privacy, your boss gives you a massive data set containing information about past customers:

- Their age
- The number of friends they have on Facebook
- The place where they currently live
- How frequently they travel
- The size of their household
- Their average monthly salary
- Their initial bid
  (assuming they used the Name Your Own Price tool)
- Gender
- Marital status
- Average time spent on the internet
- How many times they searched for the same flight
- Their favorite movie
- …

For every customer in your data set you also get to see how much they actually ended up paying for the ticket. This could be with the Name Your Own Price tool, or just using the regular travel tool of Priceline or other competitor websites.

So overall, you have access to $p+1$ vectors in $\mathbb{R}^n$:

$$x_1 = \begin{pmatrix} x_1^1 \\ x_1^2 \\ \vdots \\ x_1^n \end{pmatrix}, x_2 = \begin{pmatrix} x_2^1 \\ x_2^2 \\ \vdots \\ x_2^n \end{pmatrix}, \dots, x_p = \begin{pmatrix} x_p^1 \\ x_p^2 \\ \vdots \\ x_p^n \end{pmatrix}, \qquad y = \begin{pmatrix} y^1 \\ y^2 \\ \vdots \\ y^n \end{pmatrix},$$

where the vector x_i summarizes the i-th attribute of all customors (e.g., x_1 can have all the ages), and the vector y contains the prices that different individuals paid.

You would like to find a simple relationship between y and the attiributes x_i. For each customers j, you believe that:

$$y^j \approx \beta_0 + \beta_1 * age + \beta_2 * monthly\ salary + \cdots$$
$$\text{i.e., } y^j \approx \beta_0 + \beta_1 x_1^j + \beta_2 x_2^j + \cdots + \beta_p x_p^j, \forall j, \text{ where } \beta_0, \beta_1, \dots, \beta_p \in \mathbb{R}.$$

In vector notation, $y \approx \beta_0 \mathbf{1} + \sum_{k=1}^{p} \beta_k x_k \Leftrightarrow\ y - \beta_0 \mathbf{1} - \sum_{k=1}^{p} \beta_k x_k \approx 0$.

- The natural optimization problem to solve to find the best coefficients (β_k) is then:

$$\min_{\beta} \left\| y - \beta_0 - \sum_{k=1}^{p} \beta_k x_k \right\|_2$$

This is a convex optimization problem. Which rule would you use to see this?

- You also feel that many of the p attributes you were given are irrelevant. So you want to find a solution where many of the β_k's are zero; i.e., the behavior of the customors is explained by a few attributes only.

- Ideally, you would add a constraint of the type $\#\{\beta_k = 0\}$ is large. Unfortunately, this constraint is not convex and not an easy one to deal with computationally.

- Instead, a common approach taken in practice is to make

$$\|\beta\|_1 = |\beta_1| + \cdots + |\beta_p|$$

  as small as possible. This encourages many components of β to be zero or close to zero. You can threshold the ones that are very small down to zero.

Your new optimization problem then becomes:

$$\min_{\beta} \left\| y - \beta_0 - \sum_{k=1}^{p} \beta_k x_k \right\|_2 + \gamma \|\beta\|_1,$$

where γ is some positive constant picked by you. This is again a convex optimization problem. To see this, we are using Rule 1 and Rule 2. Do you agree?

Note that:

- If γ is big, then more effort goes into minimizing the second term.
- If γ is small, then more effort goes into minimizing the first term.
- Making the first term small makes the error in the prediction of the model on past customers small.
- Making the second term small ensures that we don't use too many attributes in our predictor.
  - This can help with making better predictions for future customers by avoiding overfitting.

In statistics, the solution of this optimization problem is called the **LASSO** estimator.

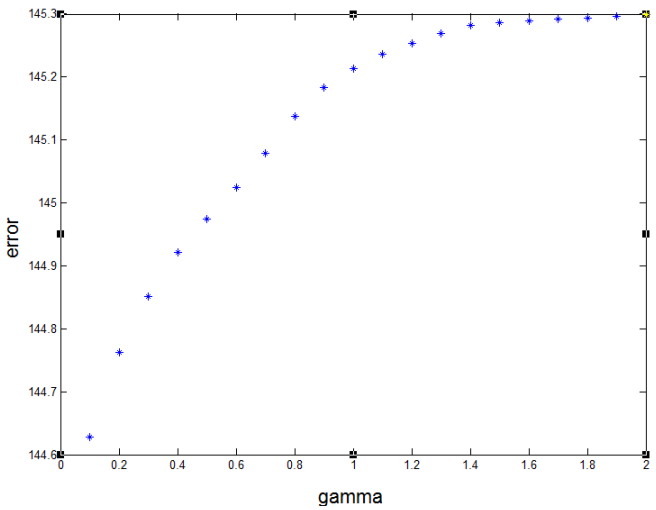
Solving the problem with CVX

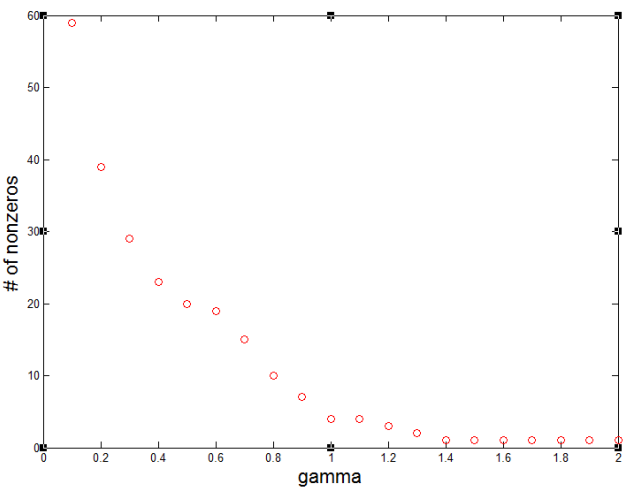
- We solve an instance of this problem with CVX with
  - $n = 10000$ (# of past customers)
  - $p = 100$ (# of attributes of each customer)
  - β: varying between 0.1 and 2

```
clear all
n=10000;
p=100;

error_vec=[];
nnzeros=[];
X=[ones(n,1) rand(n,p)];
y=5*rand(n,1);
for gamma=.1:.1:2
cvx_begin
        variable b(p+1)
        minimize (norm(y-X*b,2)+gamma*norm(b,1))
cvx_end
error_vec=[error_vec;norm(y-X*b,2)];
nnzeros=[nnzeros; length(find(abs(b)>.01))];
end
plot(.1:.1:2, error_vec,'*')
figure, plot(.1:.1:2, nnzeros,'ro')
```

- Run time is a few seconds (even on my tablet)!
- Once we choose a γ we like, we keep the vector β^* (the ouput of the optimization problem) as our predictor.
- When a new costumer comes in with a new set of attributes $x_1, \dots, x_p$, we simply predict the price he is willing to pay to be $\beta_0^* + \beta_1^* x_1 + \beta_2^* x_2 + \cdots + \beta_p^* x_p$.
- So the optimization problem is solved offline and only once. The computation done online at the time of prediction is simply taking a simple vector inner product.





A good technique for choosing γ is *cross validation*.

Application II: Support Vector Machines (SVMs)

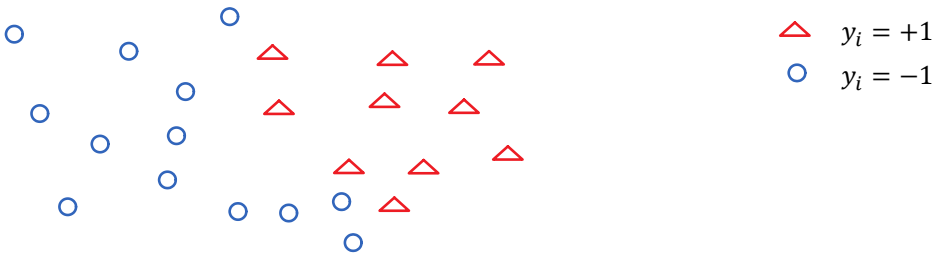
- An example of supervised learning: we would like to learn a classifier from a labeled data set (called the training set).
- The classifier is then used to label future data points.
- Classic example is an email spam filter:
  - Given a large number of emails with correct labels "spam" or "not spam", we would like an algorithm for classifying future emails as spam or not spam.
  - The emails for which we already have the labels constitute the "training set".

Example:

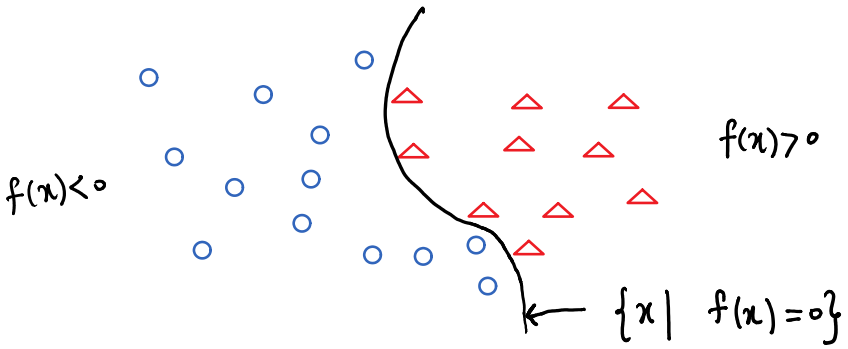
Hello class, My office hours this week have moved to Thursday, 4-5:30 PM. Lecture 4 is now up on the course website. -Amirali	Good day, My name is Chaghal. I seek true soulmate. Are you ready for relations? Check my profile here: http://soul4.com/me.exe	Hey man, I'm tired of this homework for ORF 363. Let's go party tonight. We can always ask for an extension. -J
Not spam	**Spam**	**Spam**

- A basic approach is to associate a pair (x_i, y_i) to each email; y_i is the label, which is either 1 (spam) or -1 (not spam). The vector $x_i \in \mathbb{R}^n$ is called a feature vector; it collects some relevant information about email i. For example:
  - How many words are in the email?
  - How many misspelled words?
  - How many links?
  - Is there a $ sign?
  - Does the word "bank account" appear?
  - Is the sender's email client trustworthy?
  - …

- If we have m emails, we end up with m vectors in $\mathbb{R}^n$, each with a label ± 1. Here is a toy example in $\mathbb{R}^2$:



- The goal is now to find a classifier $f: \mathbb{R}^n \to \mathbb{R}$, which takes a positive value on spam emails and a negative value on non-spam emails.
- The zero level set of f serves as a classifier for future predictions.



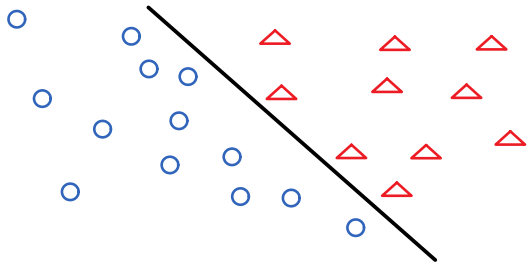
- We can search for many classes of functions as classifiers using convex optimization.
- The simplest one is *linear classification*: $f = a^T x - b$
- Need to find $a \in \mathbb{R}^n, b \in \mathbb{R}$ that satisfy

$$a^T x_i - b > 0 \text{ if } y_i = 1$$
$$a^T x_i - b < 0 \text{ if } y_i = -1$$

- This is equivalent (why?) to finding $a \in \mathbb{R}^n, b \in \mathbb{R}$ that satisfy:

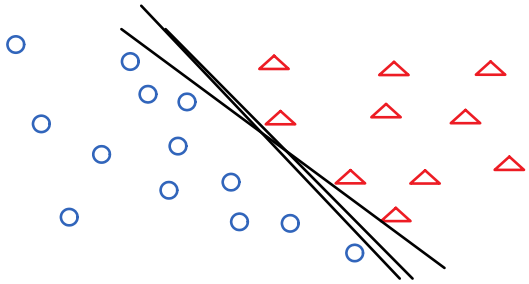
$$y_i(a^T x_i - b) \geq 1, \ i = 1, \dots, m.$$

- This is a convex feasibility problem (in fact a set of linear inequalities). It may or may not be feasible (compare examples above and below). Can you identify the condition for feasibility of linear classification?



An example of linearly separable data

- There could be many linear classifiers. Which one do we choose?



- Consider:

$$\begin{aligned} &\min_{a,b} \ ||a|| \\ &\text{s.t. } y_i(a^T x_i - b) \geq 1, \ \ i = 1, \dots, m. \end{aligned} \qquad \textbf{(1)}$$

- This is a convex program (why?)
- Optimal solution is unique (why?)
- What is this optimization problem doing?

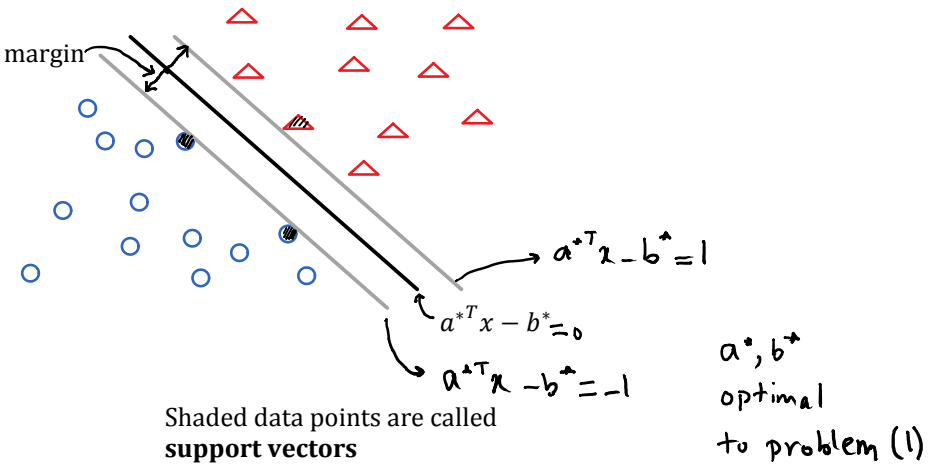
Claim 1: The optimization problem above is equivalent to:

$$\begin{aligned} &\max_{a,b,t} \ t \\ &\text{s.t. } y_i(a^T x_i - b) \geq t, \ \ i = 1, \dots, m, \\ &\quad \big|\big|a\big|\big| \leq 1. \end{aligned} \qquad \textbf{(2)}$$

Claim 2: An optimal solution of the latter problem always satisfies $\big|\big|a\big|\big| = 1$.

Claim 3: The Euclidean distance of a point $v \in \mathbb{R}^n$ to a hyperplane $a^T z = b$ is given by

$$\frac{|a^T v - b|}{||a||}.$$

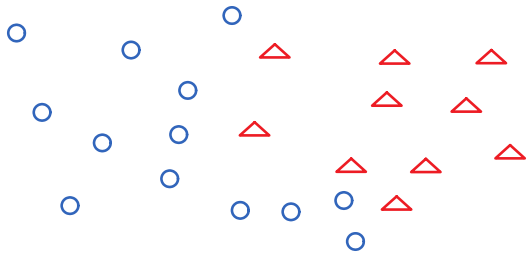


- Let's believe these three claims for the moment. What optimization problem (1) is then doing is finding a hyperplane that **maximizes the minimum distance** between the hyperplane (our classifier) and any of our data points. Do you see why?
- We are trying to buy ourselves as much margin as possible.
- This helps us be robust to noise, in case the feature vector of our future data points happen to be slightly misspecified.

Proof of the three claims: on your homework exercises! (I removed claim 3 from HW)

- Hints:
  - Claim 1: how would you get feasible solutions to one from the other?
  - Claim 2: how would you improve the objective if it didn't?
  - Claim 3: good exercise of our optimality conditions

- What if the points are not linearly separable?



- Let's try to minimize the number of points misclassified:

$$\begin{aligned} &\min_{a,b,\eta} \ ||\eta||_0 \\ &\text{s.t. } y_i(a^T x_i - b) \geq 1 - \eta_i, \ i = 1, \dots, m \\ &\quad \eta_i \geq 0, \ i = 1, \dots, m. \end{aligned}$$

- $||\eta||_0$ denotes the number of nonzero elements of η.
  - It is commonly called "the l_0 norm", but it is not really a norm (why? Which property of the norm is failing?).
- If $\eta_i = 0$, data point i is correctly classified.
- The optimization problem above is trying to set as many entries of η to zero as possible.
  - Unfortunately, it is a hard problem to solve.
  - Which entries to set to zero? Many different subsets to consider.
  - As a powerful heuristic for this problem, people solve the following:

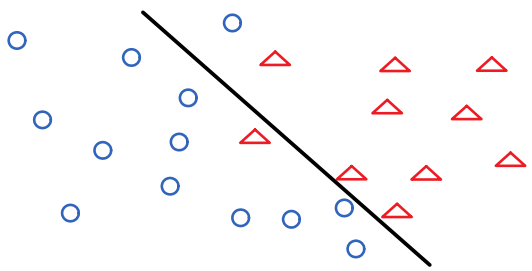
$$\begin{aligned} &\min_{a,b,\eta} \ ||\eta||_1 \\ &\text{s.t. } y_i(a^T x_i - b) \geq 1 - \eta_i, \ i = 1, \dots, m \\ &\quad \eta_i \geq 0, \ i = 1, \dots, m. \end{aligned}$$

- This is a convex program (why?). We can solve it efficiently.
- The solution with minimum l_1 norm tends to be sparse; i.e., have many entries that are zero. (There are theoretical justifications for this, beyond the scope of this class.)
- Note that when $\eta_i \leq 1$, data point i is still correctly classified, but it falls within our "margin"; hence not robustly classified.
- When $\eta_i > 1$, data point i is misclassified.

- We can solve a modified optimization problem to balance the tradeoff between the number of misclassified points and the width of our margin.

$$\min_{a,b,\eta} \ ||a|| + \gamma||\eta||_1$$
$$\text{s.t. } y_i(a^T x_i - b) \geq 1 - \eta_i, \ i = 1, \dots, m$$
$$\eta_i \geq 0, \ i = 1, \dots, m.$$

- $\gamma \geq 0$ is a parameter that we fix a priori.
- Larger γ means we assign more importance to reducing number of misclassified points.
- Smaller γ means we assign more importance to having a large margin.
  - The length of our margin is $\frac{2}{||a||}$ counting both sides (why?).
- For each γ, the problem is a convex program (why?).
- On your homework, you will run some numerical experiments on this problem.



Notes:

- Much more on convexity-preserving rules can be found in Section 3.2 of [BV04].
- You can read more about support vector machines in Section 8.6 of [BV04].
- The original LASSO paper is [Tibs96].

References:

- [AOPT13] A.A. Ahmadi, A. Olshevsky, P.A. Parrilo, and J.N. Tsitsiklis. NP-hardness of checking convexity of quartic forms and related problems. *Mathematical Programming,* 2013. http://web.mit.edu/~a_a_a/Public/Publications/convexity_nphard.pdf

- [BV04] S. Boyd and L. Vandenberghe. Convex Optimization. Cambridge University Press, 2004. http://stanford.edu/~boyd/cvxbook/

- [BG08]: S. Boyd and M. Grant. Graph implementations for nonsmooth convex programs. *Recent Advances in Learning and Control*, Springer-Verlag, 2008.

- [CVX11]: CVX Research, Inc. CVX: MATLAB software for disciplined convex programming, version 2.0. http://cvxr.com/cvx, April 2011.

- [Sche97]: J.R. Schewchuk. An introduction to the conjugate gradient method without the agonizing pain, 1994.

- [Tibs96] R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society,* 1996. http://statweb.stanford.edu/~tibs/lasso/lasso.pdf