Optimus III

CS771: Introduction to Machine Learning

Purushottam Kar

Recap of Last Lecture

Notion of *subgradient* that elegantly extends the notion of derivative to non-differentiable functions that are nevertheless convex

Subgradient calculus: scaling, sum, chain, max rules

Using the first order optimality condition to solve simple optimization problems

Using (sub)gradient descent to solve general optimization problems

Notions of *initialization, step length, convergence*

Behind the scenes in GD

$f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2 + C \cdot \sum_{i=1}^n \left[1 - y^i \cdot \mathbf{w}^\top \mathbf{x}^i\right]_+$ (ignore bias b for now)

$\nabla f(\mathbf{w}) = \mathbf{w} + C \cdot \sum_{i=1}^n g^i y^i \cdot \mathbf{x}^i$, where $g^i \in \nabla \ell_{\text{hinge}}\left(y^i \cdot \mathbf{w}^\top \mathbf{x}^i\right)$

$\mathbf{w}^{\text{new}} = \mathbf{w} - \eta \cdot \nabla f(\mathbf{w}) = (1-\eta) \cdot \mathbf{w} - \eta C \cdot \sum_{i=1}^n g^i y^i \cdot \mathbf{x}^i$

Assume $n = 1$ for a moment for sake of understanding

$\mathbf{w}^{\text{new}} = (1-\eta) \cdot \mathbf{w} - \eta C \cdot g^1 y^1 \cdot \mathbf{x}^1$

Small η: $(1-\eta)$ is large $\Rightarrow$ do not change $\mathbf{w}$ too much!

Large η: Feel free to change $\mathbf{w}$ as much as the gradient dictates

If $\mathbf{w}$ does well on $(\boldsymbol{x}^1, y^1)$, say $y^1 \cdot \mathbf{w}^\top \boldsymbol{x}^1 > 1$, then $g^1 = 0$

If $\mathbf{w}$ does badly on $(\boldsymbol{x}^1, y^1)$, say $y^1 \cdot \mathbf{w}^\top \boldsymbol{x}^1 < 0$, then $g^1 = -1$

$\mathbf{w}^{\text{new}} = (1-\eta) \cdot \mathbf{w} + \eta C \cdot y^1 \cdot \mathbf{x}^1$

$y^1 \cdot (\mathbf{w}^{\text{new}})^\top \cdot \mathbf{x}^1 = (1-\eta) y^1 \cdot \mathbf{w}^\top \boldsymbol{x}^1 + \eta C \cdot \|\mathbf{x}^1\|_2^2$

Behind the sce

So gradient descent, although a mathematical tool from calculus, actually tries very actively to make the model perform better on all data points

$f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2 + C \cdot \sum$ … $]_+$ … now)

$\nabla f(\mathbf{w}) = \mathbf{w} + C \cdot \sum_{i=1}^n g^i y^i \cdot \mathbf{x}^i$, where $g^i \in \nabla \ell_{\text{hinge}}(y^i \cdot \mathbf{w}^\top \mathbf{x}^i)$

$\mathbf{w}^{\text{new}} = \mathbf{w} - \eta \cdot \nabla f(\mathbf{w}) = (1-\eta) \cdot \mathbf{w} - \eta C \cdot \sum_{i=1}^n g^i y^i \cdot \mathbf{x}^i$

Assume $n = 1$ for a moment for sake of understanding

$\mathbf{w}^{\text{new}} = (1-\eta) \cdot \mathbf{w} - \eta C \cdot g^1 y^1 \cdot \mathbf{x}^1$

Small η: $(1-\eta)$ is large $\Rightarrow$ do not change $\mathbf{w}$ too much

Large η: Feel free to change $\mathbf{w}$ as much as the gradient dictates

If $\mathbf{w}$ does well on $(\boldsymbol{x}^1, y^1)$, say $y^1 \cdot \mathbf{w}^\top \boldsymbol{x}^1 > 1$, then $g^1 = 0$

If $\mathbf{w}$ does badly on $(\boldsymbol{x}^1, y^1)$, say $y^1 \cdot \mathbf{w}^\top \boldsymbol{x}^1 < 0$, then g

$\mathbf{w}^{\text{new}} = (1-\eta) \cdot \mathbf{w} + \eta C \cdot y^1 \cdot \mathbf{x}^1$

$y^1 \cdot (\mathbf{w}^{\text{new}})^\top \cdot \mathbf{x}^1 = (1-\eta) y^1 \cdot \mathbf{w}^\top \boldsymbol{x}^1 + \eta C \cdot \|\mathbf{x}^1\|_2^2$

No change to $\mathbf{w}$ due to the data point $(\boldsymbol{x}^1, y^1)$

$\mathbf{w}^{\text{new}}$ may get much better margin on $(\boldsymbol{x}^1, y^1)$ than $\mathbf{w}$

Stochastic Gradient Method

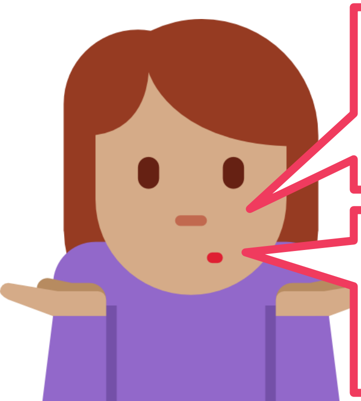
$\nabla f(\mathbf{w}) = \mathbf{w} + C \cdot \sum_{i=1}^{n} g^i y^i \cdot \mathbf{x}^i$, where $g^i \in \nabla \ell_{\text{hinge}}\left(y^i \cdot \mathbf{w}^\top \mathbf{x}^i\right)$

Calculating each g^i takes $\mathcal{O}(d)$ time since $\mathbf{w}, \mathbf{x}^i \in \mathbb{R}^d$ - total $\mathcal{O}(nd)$

At each time, choose a random data point $\left(\mathbf{x}^{i_t}, y^{i_t}\right)$
$\nabla f(\mathbf{w}) \approx \mathbf{w} + C \cdot g^{i_t} y^{i_t} \cdot \mathbf{x}^{i_t}$ - only $\mathcal{O}(d)$ time!!

Warning: may have to perform several SGD steps than we had to do with GD but each SGD step is much cheaper than a GD step

We take a random data point to avoid being unlucky (also it is cheap)

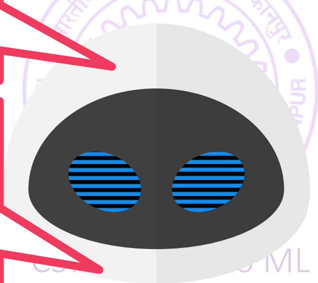


Do we really need to spend so much time on just one update?

Initially, all we need is a general direction in which to move

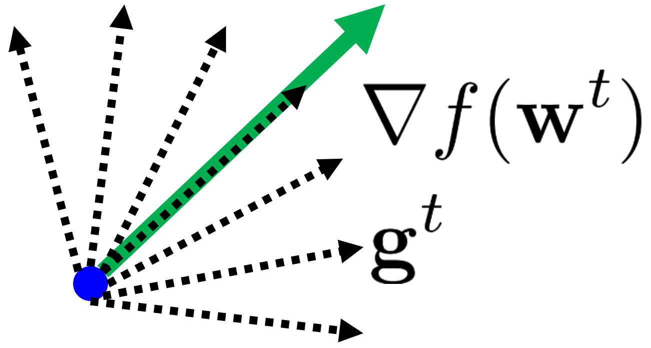
Especially in the beginning, when we are far away from the optimum!

No, SGD gives a cheaper way to perform gradient descent



Mini-batch SGD

If data is very diverse, the “stochastic” gradient may vary quite a lot depending on which random data point is chosen



This is called *variance* (more on this later) but this can slow down the SGD process – make it jittery

One solution, choose more than one random point

At each step, choose B random data points (B = *mini batch size)* without replacement, say $\left(x^{i_t^1}, y^{i_t^1}\right), \ldots, \left(x^{i_t^B}, y^{i_t^B}\right)$ and use

$$\nabla f(\mathbf{w}) \approx \mathbf{w} + C \cdot \sum_{b=1}^{B} g^{i_t^b} y^{i_t^b} \cdot \mathbf{x}^{i_t^b}$$

Takes $\mathcal{O}(Bd)$ time to execute MBSGD – more expensive than SGD

Notice that if $B = n$ then MBSGD becomes plain GD

Constrained Optimization

Recall that an optimization problem has an objective and constraints

$\min_x f(x)$

such that $p(x) < 0$

and $q(x) > 0$ etc.

Objective

Constraints

The set of points that satisfy *all* the constraints is called the *feasible set* $\mathcal{C}$

$$\mathcal{C} \triangleq \{x : p(x) < 0 \text{ and } q(x) > 0 \text{ and } \ldots.\}$$

Problems with constraints more challenging

Method 1: Interior Point Method

Find a way to initialize within $\mathcal{C}$ and then take steps that never go out

A very powerful family of methods – also very involved

Not extremely popular in machine learning – can be expensive

Beyond the scope of CS771

Constrained Optimization

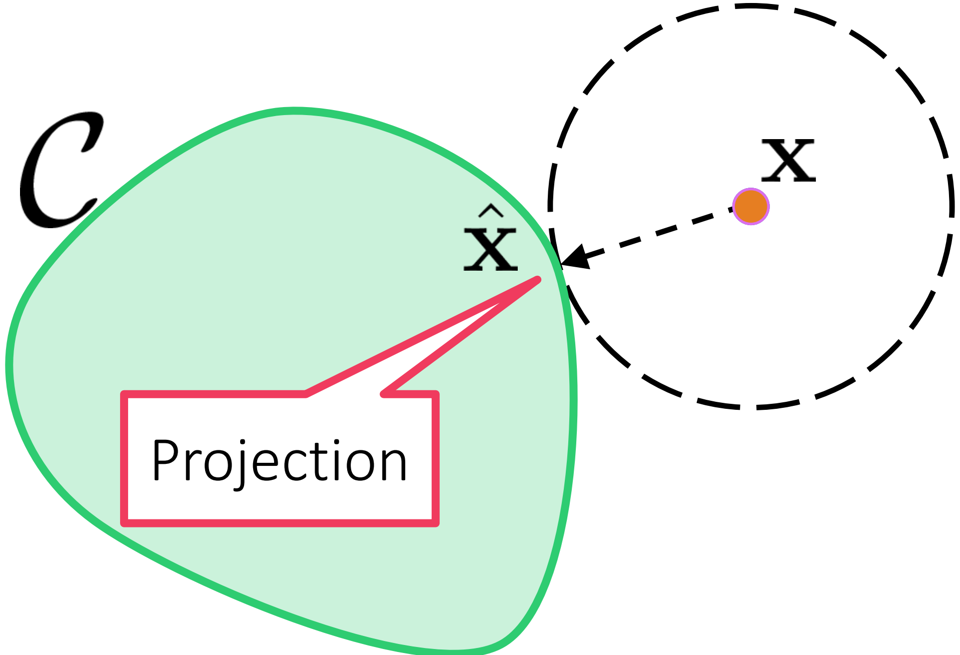
Method 2: Projected Gradient Descent

Perform (stochastic) gradient descent as usual. However, if this causes us to step outside the feasible set $\mathcal{C}$, go back to the feasible set

Process of "going back" into $\mathcal{C}$: projection step

$$\hat{\mathbf{x}} = \Pi_{\mathcal{C}}(\mathbf{x}) = \arg\min_{\mathbf{z}\in\mathcal{C}} \|\mathbf{x} - \mathbf{z}\|_2^2$$

***Warning**: works only if $\mathcal{C}$ is such that projection step is easy*



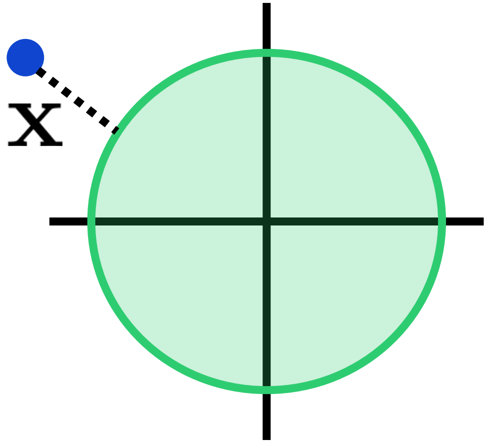
PROJECTED (SUB)GRADIENT DESCENT

1. Choose $\mathbf{g}^t \in \partial f(\mathbf{w}^t)$, step length η_t
2. Update $\mathbf{u}^{t+1} \leftarrow \mathbf{w}^t - \eta_t \cdot \mathbf{g}^t$
3. Project $\mathbf{w}^{t+1} \leftarrow \Pi_{\mathcal{C}}(\mathbf{u}^{t+1})$
4. Repeat until convergence

A few useful Projections

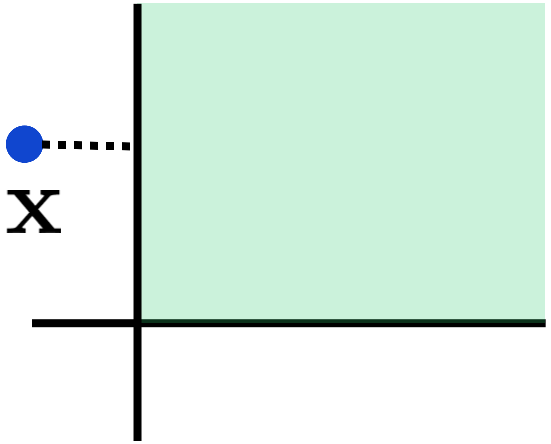
$$\hat{\mathbf{x}} = \Pi_{\mathcal{C}}(\mathbf{x})$$

$$\mathcal{C} = \{\mathbf{x} : \|\mathbf{x}\|_2 \leq 1\}$$



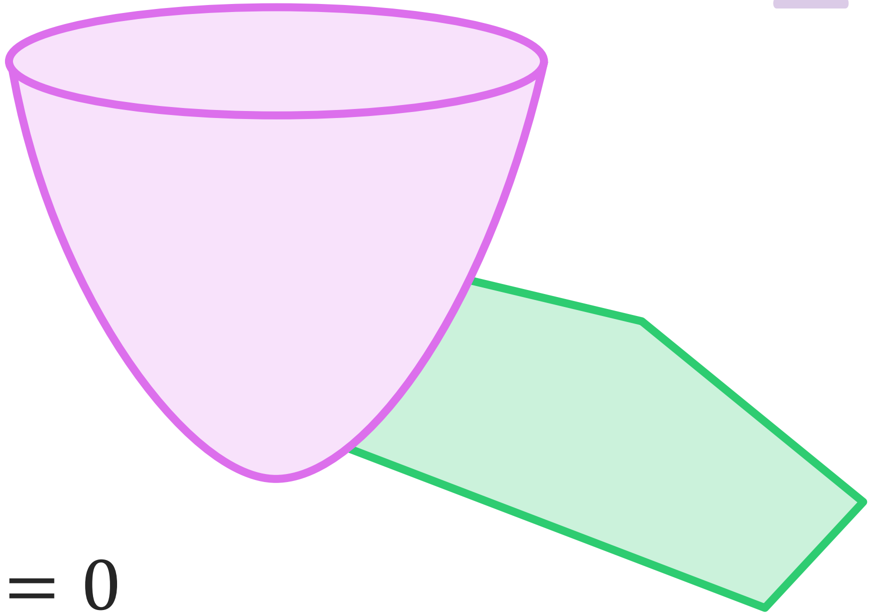
$$\hat{\mathbf{x}} = \begin{cases} \mathbf{x} & \text{if } \|\mathbf{x}\|_2 \leq 1 \\ \frac{\mathbf{x}}{\|\mathbf{x}\|_2} & \text{if } \|\mathbf{x}\|_2 > 1 \end{cases}$$

$$\mathcal{C} = \{\mathbf{x} : \mathbf{x}_i \geq 0\}$$



$$\hat{\mathbf{x}}_i = \begin{cases} \mathbf{x}_i & \text{if } \mathbf{x}_i \geq 0 \\ 0 & \text{if } \mathbf{x}_i < 0 \end{cases}$$

Constrained Optimization



Method 3: Creating a Dual Problem

Suppose we wish to solve

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x})$$

s.t. $g(\mathbf{x}) \leq 0$

***Trick**: sneak this constraint into the objective*

Construct a barrier function $r(\mathbf{x})$ *so that* $r(\mathbf{x}) = 0$ *if* $g(\mathbf{x}) \leq 0$ *and* $r(\mathbf{x}) = \infty$ *otherwise, and simply solve*

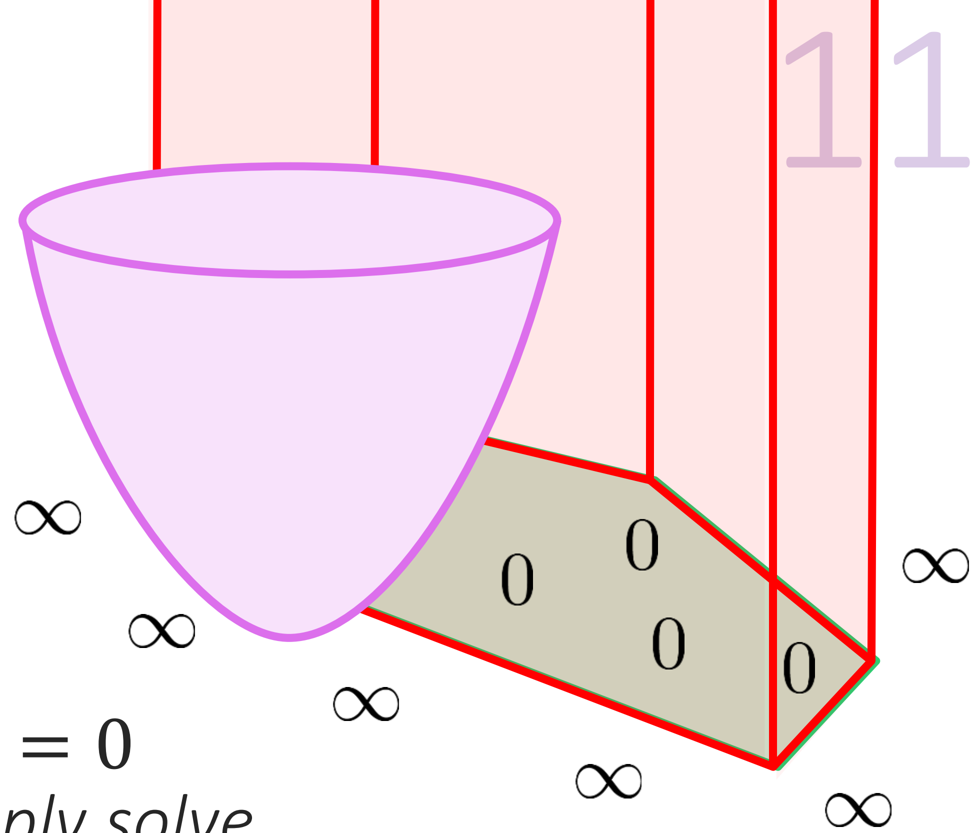
Easy to see that both problems have the same solution

One very elegant way to construct such a barrier is the following

$$r(\mathbf{x}) = \max_{\alpha\geq 0} \alpha \cdot g(\mathbf{x})$$

Thus, we want to solve $\min_{\mathbf{x}\in\mathbb{R}^d} \left\{ f(\mathbf{x}) + \max_{\alpha\geq 0} \alpha \cdot g(\mathbf{x}) \right\}$

Constrained Optimization



Method 3: Creating a Dual Problem

Suppose we wish to solve

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) + r(\mathbf{x})$$

***Trick**: sneak this constraint into the objective*

Construct a barrier function $r(\mathbf{x})$ *so that* $r(\mathbf{x}) = 0$ *if* $g(\mathbf{x}) \leq 0$ *and* $r(\mathbf{x}) = \infty$ *otherwise, and simply solve*

Easy to see that both problems have the same solution

One very elegant way to construct such a barrier is the following

$$r(\mathbf{x}) = \max_{\alpha\geq 0} \alpha \cdot g(\mathbf{x})$$

Thus, we want to solve $\min_{\mathbf{x}\in\mathbb{R}^d} \left\{ f(\mathbf{x}) + \max_{\alpha\geq 0} \alpha \cdot g(\mathbf{x}) \right\}$

Constrained Optimization

Method 3: Creating a Dual Problem

Suppose we wish to solve

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) + r(\mathbf{x})$$

***Trick**: sneak this constraint into the objective*

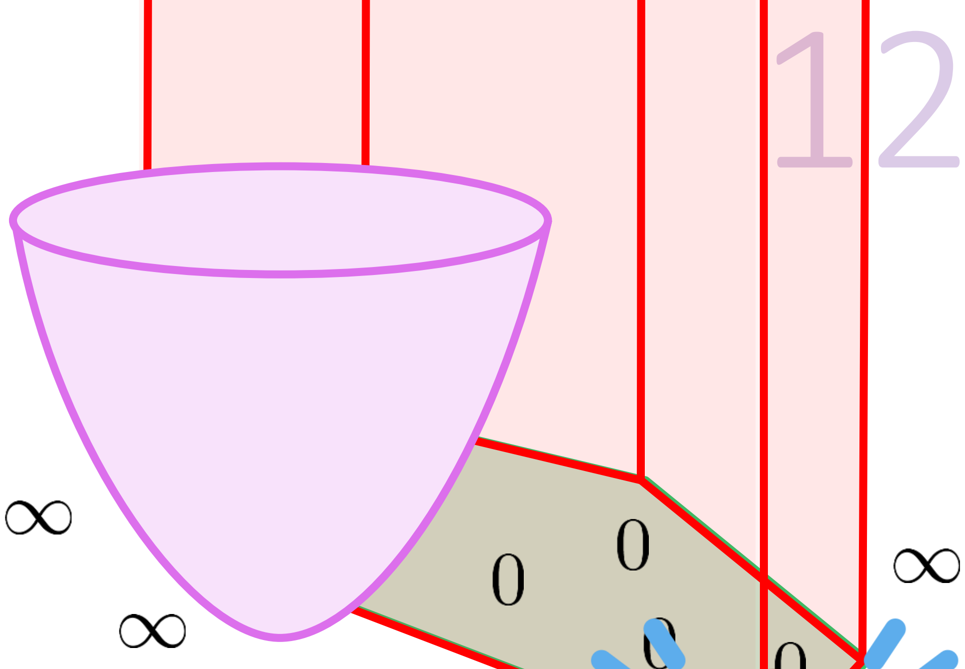
Construct a barrier function $r(\mathbf{x})$ so that $r(\mathbf{x}) = 0$

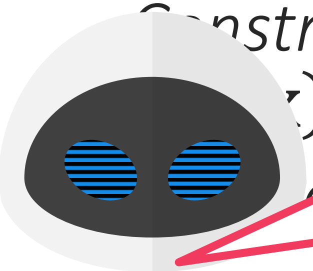
$\mathbf{x})$ … wise, an

… ave the

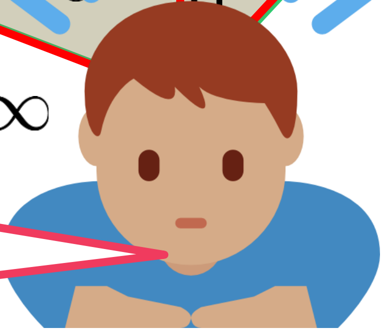
One very elegant way to construct such a barrier is the following

Thus, we want to so





Let us see how to handle multiple constraints and equality constraints



Hmm … we still have a constraint here, but a very simple one i.e. $\alpha \geq 0$

Same as $\min_{\mathbf{x}\in\mathbb{R}^d} \left\{ \max_{\alpha\geq 0} \{ f(\mathbf{x}) + \alpha \cdot g(\mathbf{x}) \} \right\}$

A few Cleanup Steps

Step 1: Convert your problem to a minimization problem

$$\max f(\mathbf{x}) \rightarrow \min -f(\mathbf{x})$$

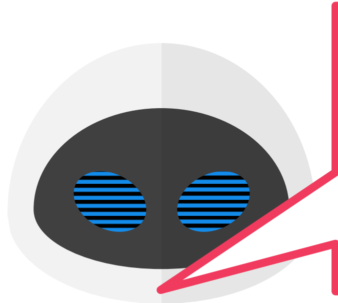
Step 2: Convert all inequality constraints to $\leq$ constraints

$$g(\mathbf{x}) \geq 0 \rightarrow -g(\mathbf{x}) \leq 0$$

Step 3: Convert all equality constraints to two inequality constraints

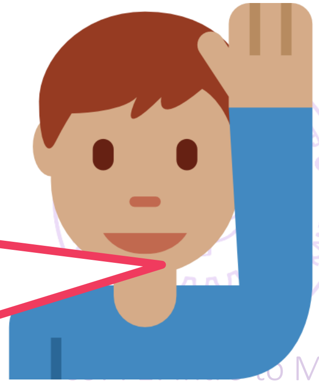
$$s(\mathbf{x}) = 0 \rightarrow s(\mathbf{x}) \leq 0, -s(\mathbf{x}) \leq 0$$

Step 4: For each constraint we now have, introduce a new variable e.g. if we have C inequality constraints $g_1(\mathbf{x}) \leq 0, \ldots, g_C(\mathbf{x}) \leq 0$, introduce C new variables $\alpha_1, \ldots, \alpha_C$



The variables of the original optimization problem, e.g. $\mathbf{x}$ in this case, are called the *primal variables* by comparison

These new variables are called *dual variables* or sometimes even called *Lagrange multipliers*



The Lagrangian

$$\min_{\mathbf{x}} f(\mathbf{x})$$

$$\text{s.t. } g_1(\mathbf{x}) \leq 0$$
$$g_2(\mathbf{x}) \leq 0$$
$$\vdots$$
$$g_C(\mathbf{x}) \leq 0$$

$\mathcal{L}(\mathbf{x}, \boldsymbol{\alpha}) = f(\mathbf{x}) + \sum_{c=1}^{C} \boldsymbol{\alpha}_c \cdot g_c(\mathbf{x})$ called the *Lagrangian* of the problem

If $\mathbf{x}$ violates even one constraint, we have

$$\max_{\substack{\boldsymbol{\alpha} \in \mathbb{R}^C \\ \boldsymbol{\alpha}_c \geq 0}} \{\mathcal{L}(\mathbf{x}, \boldsymbol{\alpha})\} = \infty$$

If $\mathbf{x}$ satisfies every single constraint, we have

$$\max_{\substack{\boldsymbol{\alpha} \in \mathbb{R}^C \\ \boldsymbol{\alpha}_c \geq 0}} \{\mathcal{L}(\mathbf{x}, \boldsymbol{\alpha})\} = f(\mathbf{x})$$

$$\min_{\mathbf{x} \in \mathbb{R}^d} \left\{ \max_{\substack{\boldsymbol{\alpha} \in \mathbb{R}^C \\ \boldsymbol{\alpha}_c \geq 0}} \left\{ f(\mathbf{x}) + \sum_{c=1}^{C} \boldsymbol{\alpha}_c \cdot g_c(\mathbf{x}) \right\} \right\}$$

The Lagrangian

$$\min_{\mathbf{x}} f(\mathbf{x})$$

$$\text{s.t. } g_1(\mathbf{x}) \le 0$$
$$g_2(\mathbf{x}) \le 0$$
$$\vdots$$
$$g_C(\mathbf{x}) \le 0$$

$\mathcal{L}(\mathbf{x}, \boldsymbol{\alpha}) = f(\mathbf{x}) + \sum_{c=1}^{C} \boldsymbol{\alpha}_c \cdot g_c(\mathbf{x})$ called the *Lagrangian* of the problem

If $\mathbf{x}$ violates even one constraint, we have

$$\max_{\substack{\boldsymbol{\alpha} \in \mathbb{R}^C \\ \boldsymbol{\alpha}_c \ge 0}} \{\mathcal{L}(\mathbf{x}, \boldsymbol{\alpha})\} = \infty$$

If $\mathbf{x}$ satisfies every single constraint, we have

This is just a nice way of rewriting the above problem

$$\min_{\mathbf{x} \in \mathbb{R}^d} \left\{ \max_{\substack{\boldsymbol{\alpha} \in \mathbb{R}^C \\ \boldsymbol{\alpha}_c \ge 0}} \left\{ f(\mathbf{x}) + \sum_{c=1}^{C} \boldsymbol{\alpha}_c \cdot g_c(\mathbf{x}) \right\} \right\}$$

The Dual Problem

The original optimization problem is also called the *primal problem*

Recall: variables of the original problem e.g. $\mathbf{x}$ called *primal variables*

Using the Lagrangian, we rewrote the primal problem as

$$\min_{\mathbf{x}\in\mathbb{R}^d} \left\{ \max_{\substack{\boldsymbol{\alpha}\in\mathbb{R}^C \\ \boldsymbol{\alpha}_c\geq 0}} \left\{ f(\mathbf{x}) + \sum_{c=1}^{C} \boldsymbol{\alpha}_c \cdot g_c(\mathbf{x}) \right\} \right\}$$

The dual problem is obtained by simply switching order of min/max

$$\max_{\substack{\boldsymbol{\alpha}\in\mathbb{R}^C \\ \boldsymbol{\alpha}_c\geq 0}} \left\{ \min_{\mathbf{x}\in\mathbb{R}^d} \left\{ f(\mathbf{x}) + \sum_{c=1}^{C} \boldsymbol{\alpha}_c \cdot g_c(\mathbf{x}) \right\} \right\}$$

In some cases, the dual problem is easier to solve than the primal

Duality

Let $\hat{\mathbf{x}}^P, \hat{\boldsymbol{\alpha}}^P$ be the solutions to the primal problem i.e.

$$(\hat{\mathbf{x}}^P, \hat{\boldsymbol{\alpha}}^P) = \underset{\mathbf{x}\in\mathbb{R}^d}{\operatorname{argmin}} \left\{ \underset{\substack{\boldsymbol{\alpha}\in\mathbb{R}^C \\ \boldsymbol{\alpha}_c\geq 0}}{\operatorname{argmax}} \left\{ f(\mathbf{x}) + \sum_{c=1}^{C} \boldsymbol{\alpha}_c \cdot g_c(\mathbf{x}) \right\} \right\}$$

Let $\hat{\mathbf{x}}^D, \hat{\boldsymbol{\alpha}}^D$ be the solutions to the dual problem i.e.

$$(\hat{\mathbf{x}}^D, \hat{\boldsymbol{\alpha}}^D) = \underset{\substack{\boldsymbol{\alpha}\in\mathbb{R}^C \\ \boldsymbol{\alpha}_c\geq 0}}{\operatorname{argmax}} \left\{ \underset{\mathbf{x}\in\mathbb{R}^d}{\operatorname{argmin}} \left\{ f(\mathbf{x}) + \sum_{c=1}^{C} \boldsymbol{\alpha}_c \cdot g_c(\mathbf{x}) \right\} \right\}$$

Strong Duality: $\hat{\mathbf{x}}^P = \hat{\mathbf{x}}^D$ if the original problem is convex and “nice”

Complementary Slackness: $\hat{\boldsymbol{\alpha}}_c^D \cdot g_c(\hat{\mathbf{x}}^D) = 0$ for all constraints c

Note: not complimentary but complementary ☺

Hard SVM without a bias

$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{w}\|_2^2$ such that $1 - y^i \cdot \mathbf{w}^\top \mathbf{x}^i \leq 0$ for all $i \in [n]$

n constraints so we need n dual variables i.e. $\boldsymbol{\alpha} \in \mathbb{R}^n$

Lagrangian: $\mathcal{L}(\mathbf{w}, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|_2^2 + \sum_{i=1}^n \boldsymbol{\alpha}_i\left(1 - y^i \cdot \mathbf{w}^\top \mathbf{x}^i\right)$

Primal problem: $\underset{\mathbf{w}\in\mathbb{R}^d}{\operatorname{argmin}} \left\{\underset{\boldsymbol{\alpha}\geq 0}{\operatorname{argmax}} \left\{\frac{1}{2}\|\mathbf{w}\|_2^2 + \sum_{i=1}^n \boldsymbol{\alpha}_i\left(1 - y^i \mathbf{w}^\top \mathbf{x}^i\right)\right\}\right\}$

Dual problem: $\underset{\boldsymbol{\alpha}\geq 0}{\operatorname{argmax}} \left\{\underset{\mathbf{w}\in\mathbb{R}^d}{\operatorname{argmin}} \left\{\frac{1}{2}\|\mathbf{w}\|_2^2 + \sum_{i=1}^n \boldsymbol{\alpha}_i\left(1 - y^i \mathbf{w}^\top \mathbf{x}^i\right)\right\}\right\}$

The dual problem can be greatly simplified!

Simplifying the Dual Problem

Note that the inner problem in the dual problem is

$$\underset{\boldsymbol{\alpha}\geq 0}{\arg\max}\left\{\underset{\mathbf{w}\in\mathbb{R}^d}{\arg\min}\left\{\frac{1}{2}\|\mathbf{w}\|_2^2+\sum_{i=1}^n \boldsymbol{\alpha}_i\left(1-y^i\mathbf{w}^\top\mathbf{x}^i\right)\right\}\right\}$$

Since this is an unconstrained problem with a convex and differentiable objective, we can apply first order optimality to solve it completely ☺

If we set the gradient to zero, we will get $\mathbf{w}=\sum_{i=1}^n \boldsymbol{\alpha}_i y^i\cdot\mathbf{x}^i$

Substituting this back in the dual problem we get

$$\underset{\boldsymbol{\alpha}\geq 0}{\arg\max}\left\{\sum_{i=1}^n \boldsymbol{\alpha}_i-\frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n \boldsymbol{\alpha}_i\boldsymbol{\alpha}_j y^i y^j\left\langle\mathbf{x}^i,\mathbf{x}^j\right\rangle\right\}$$

This is actually the problem several solvers (e.g. libsvm, sklearn) solve

Simplifying the Dual Prob

Once you get optimal values of $\boldsymbol{\alpha}$, use $\mathbf{w} = \sum_{i=1}^{n} \boldsymbol{\alpha}_i y^i \cdot \mathbf{x}^i$ to get optimal value of $\mathbf{w}$

Note that the inner problem in the dual problem is

$$\underset{\boldsymbol{\alpha}\geq 0}{\operatorname{argmax}} \left\{ \underset{\mathbf{w}\in\mathbb{R}^d}{\operatorname{argmin}} \left\{ \frac{1}{2}\|\mathbf{w}\|_2^2 + \sum_{i=1}^{n} \boldsymbol{\alpha}_i \left(1 - y^i \mathbf{w}^\top \mathbf{x}^i\right) \right\} \right\}$$

Since this is an unconstrained problem with a convex and differentiable objective, we can apply first order optimality to solve it completely ☺

If we set the gradient to zero, we will get $\mathbf{w} = \sum_{i=1}^{n} \boldsymbol{\alpha}_i y^i \cdot \mathbf{x}^i$

Substituting this back in the dual problem we get

$$\underset{\boldsymbol{\alpha}\geq 0}{\operatorname{argmax}} \left\{ \sum_{i=1}^{n} \boldsymbol{\alpha}_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j y^i y^j \left\langle \mathbf{x}^i, \mathbf{x}^j \right\rangle \right\}$$

This is actually the problem several solvers (e.g. libsvm, sklearn) solve

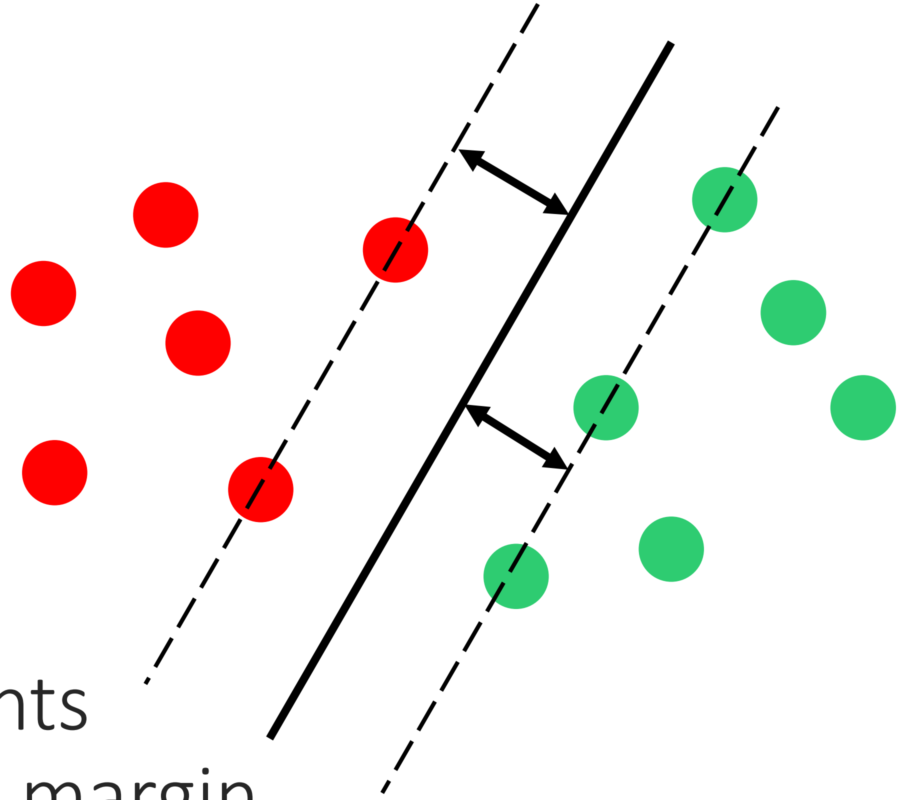
Support Vectors

Recall: we have $\boldsymbol{\alpha}_i$ for every data point

After solving the dual problem, the data points for which $\boldsymbol{\alpha}_i \neq 0$: **Support Vectors**

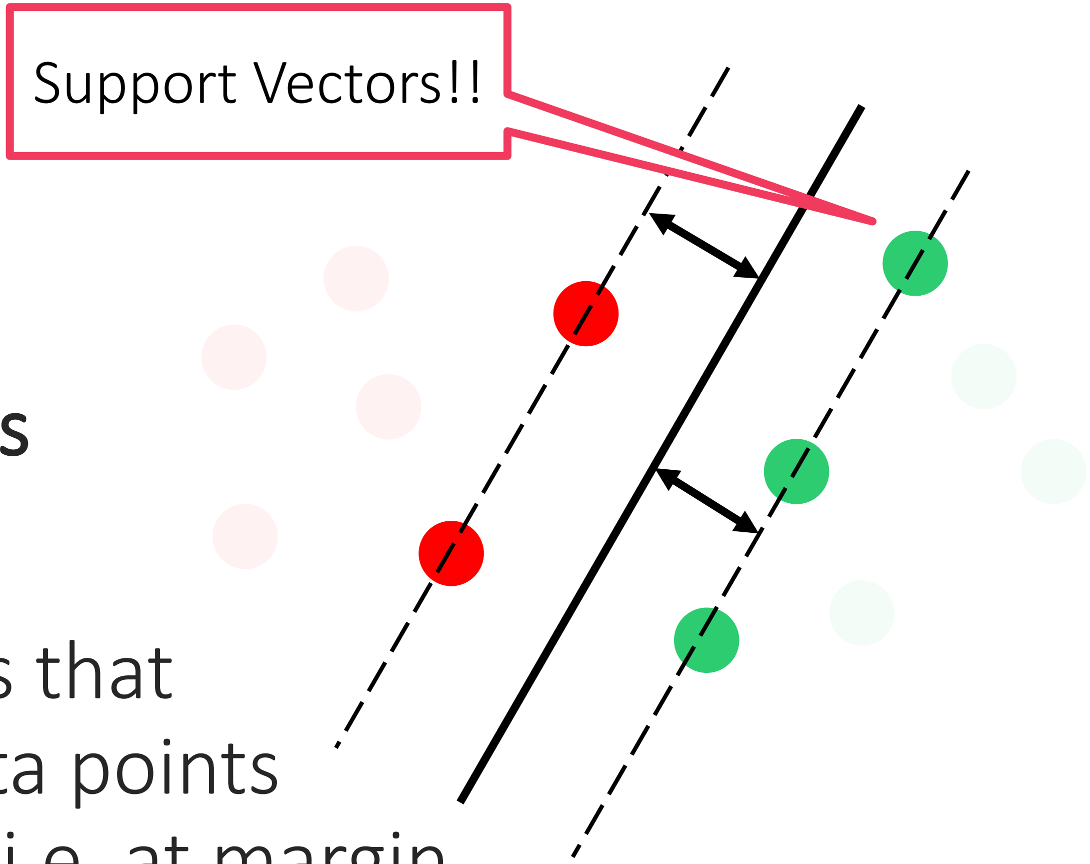
Usually we have $\ll n$ support vectors

Recall: complementary slackness tells us that $\boldsymbol{\alpha}_i\left(1 - y^i \mathbf{w}^\top \mathbf{x}^i\right) = 0$ i.e. only those data points can become SVs for which $y^i \mathbf{w}^\top \mathbf{x}^i = 1$ i.e. at margin



The reason these are called *support* vectors has to do with a mechanical interpretation of these objects – need to look at CSVM to understand that

Support Vectors



Recall: we have $\boldsymbol{\alpha}_i$ for every data point

After solving the dual problem, the data points for which $\boldsymbol{\alpha}_i \neq 0$: **Support Vectors**

Usually we have $\ll n$ support vectors

Recall: complementary slackness tells us that $\boldsymbol{\alpha}_i\left(1 - y^i \mathbf{w}^\top \mathbf{x}^i\right) = 0$ i.e. only those data points can become SVs for which $y^i \mathbf{w}^\top \mathbf{x}^i = 1$ i.e. at margin

The reason these are called *support* vectors has to do with a mechanical interpretation of these objects – need to look at CSVM to understand that

Dual for CSVM

Similar calculations show that if we have a bias term b as well as slack variables, then the dual looks like

$$\underset{\boldsymbol{\alpha}\in\mathbb{R}^n}{\operatorname{argmax}} \left\{\sum_{i=1}^n \boldsymbol{\alpha}_i - \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \boldsymbol{\alpha}_i\boldsymbol{\alpha}_j y^i y^j \left\langle \mathbf{x}^i, \mathbf{x}^j \right\rangle\right\}$$

s.t. $\boldsymbol{\alpha}_i \in [0, C]$, and $\sum_{i=1}^n \boldsymbol{\alpha}_i\, y^i = 0$

If we think of α^i as the “force” that data point $\boldsymbol{i}$ applies on the hyperplane $\mathbf{w}$ and y^i as the direction in which that force is applied, then we have total force zero since $\sum_{i=1}^n \boldsymbol{\alpha}_i\, y^i = 0$. Also since we also have $\mathbf{w} = \sum_{i=1}^n \boldsymbol{\alpha}_i y^i \cdot \mathbf{x}^i$, we can think of the torque on the hyperplane as being zero as well. Thus, the support vectors *mechanically support* the hyperplane, hence their name ☺

Next class: how to solve the dual problem?