

CE 395 Special Topics in Machine Learning

Assoc. Prof. Dr. Yuriy Mishchenko

Fall 2017

BASICS OF CONSTRAINED OPTIMIZATION

Numerical constrained optimization

- Generally, constrained optimization in KKT form is exponentially hard because it requires checking all possible combinations of $\mu_i > 0 \mid \mu_i = 0$ complementarity conditions, as we saw in the example, resulting in 2^r such checks
- Numerical solution of constrained optimization problems is nowadays done via the **interior point methods** (Karmakar, 1984)
- Interior Point methods work around the above problem by solving the optimization inside constraint interior $g(x) < -\varepsilon$, gradually taking $\varepsilon \rightarrow 0$

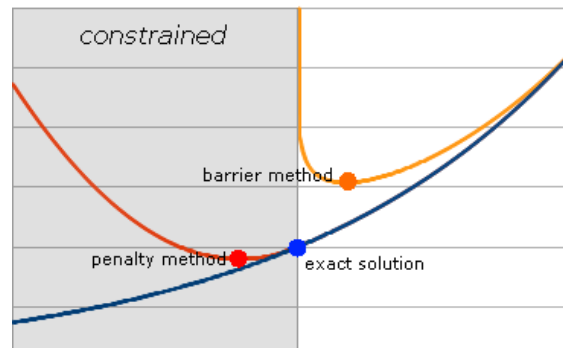
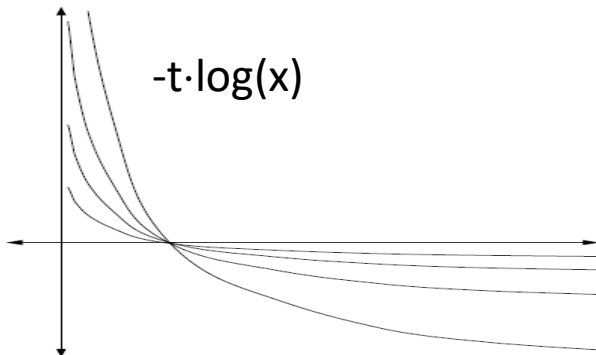
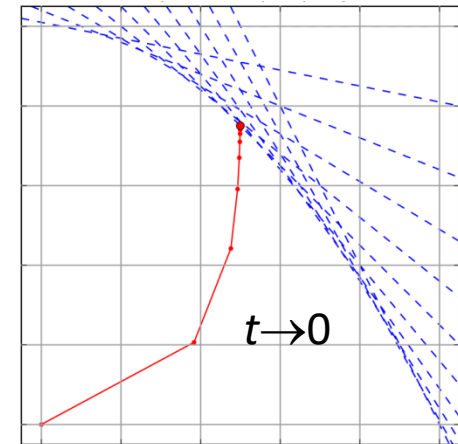
Numerical constrained optimization

Log-barrier function:

$$\min f(x) \quad s.t. \quad g_i(x) \leq 0, \quad i = 1..r$$



$$\min L_t(x), \quad L_t(x) = f(x) - t \sum_{i=1}^r \log(-g_i(x))$$



Numerical constrained optimization

- In log-barrier method, the objective function is modified by addition of **the log-barrier function** $t \cdot \sum \ln(-g_i(\mathbf{x}))$. Thus formed new objective is **unconstrained optimized** by standard methods such as GD or NM.
- Because $\ln 0 = -\infty$, the log-barrier makes it impossible for solution to cross $g_i(\mathbf{x})=0$ boundary, effectively restricting the search to the **interior of the constraint** $g_i(\mathbf{x}) < 0$.
- If the real solution is on the boundary, it is pushed inwards the search region by the barrier function by an amount proportional to the parameter t , as is illustrated in the second figure in last slide.

Numerical constrained optimization

The log-barrier method:

- Obtain the solution to $\min L_t(x)$ first with a larger value of t , let this solution be x_0
- Decrease the value of t by a constant factor, $t \rightarrow at$, and re-minimize $L_t(x)$ again with new value of t , **using the previous solution x_0 as the starting point** for minimization; next solution is x_1
- Repeat the above step by shrinking t towards zero and re-minimizing $L_t(x)$ starting from the last solution x_k , until a satisfactory answer is achieved

Duality theory

The dual function for an optimization problem $(\min f(x), g_i(x) \leq 0, h_i(x) = 0)$ is defined as :

$$G(\lambda, \mu) = \min_{any\ x} L(x, \lambda, \mu) = \min_{any\ x} \left[f(x) + \sum_{i=1}^q \lambda_i h_i(x) + \sum_{i=1}^r \mu_i g_i(x) \right]$$

*The dual function is a **function of Lagrange multipliers μ and λ** in which x is unconstrained-optimized over.*

Duality theory

Dual function has following interesting properties:

- Let $f^* = \{ \min f(x) \text{ s.t. } g_i(x) \leq 0 \text{ and } h_i(x) = 0 \}$ be the solution of the original optimization problem. **Then $G(\lambda, \mu) \leq f^*$ for any $\mu \geq 0$.**
- By extension, if $G^* = \{ \max G(\lambda, \mu) \text{ s.t. } \mu \geq 0 \}$, then **$G^* \leq f^*$** , called **duality property**.
- For many problems furthermore **$G^* = f^*$** , called **strong duality property**.

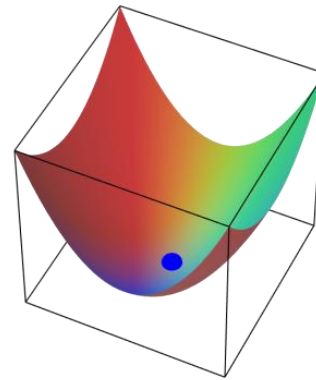
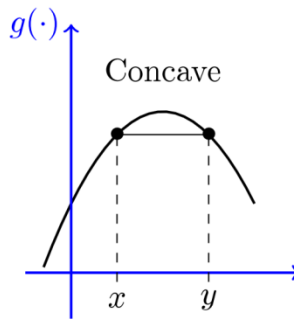
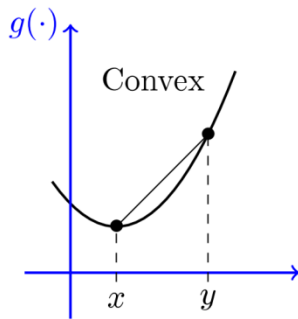
Duality theory

- The optimization problem $G^* = \{\max G(\lambda, \mu) \text{ s.t. } \mu \geq 0\}$ is called the **dual problem** (the original optimization problem is respectively known as **primal problem**).
- If strong duality holds, $G^* = f^*$, certain optimization problems are much simpler to solve via their dual counterparts (you can find a few examples in Advanced section).
- Even if strong duality is not known to hold, $G^* = f^*$ can be checked afterwards in much simpler way than solving the original problem outright.

Convex optimization

Convex functions:

$$f(\alpha x + \beta y) \leq \alpha f(x) + \beta f(y), \alpha + \beta = 1$$

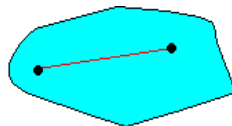


Convex function can have only one minimum \Rightarrow **any minimum found therefore is a global minimum – a very valuable property !**

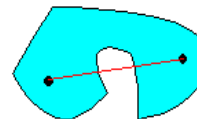
Some properties of convex functions

- If $f(x)$ is convex function $\Rightarrow f(x) \leq 0$ is **convex region**
- $f(x)$ is convex \Leftrightarrow convex along all lines $f(x_0 + t \cdot p)$
- If $f(x)$ and $g(x)$ are convex $\Rightarrow \alpha_1 g(x) + \alpha_2 f(x)$ is convex for any $\alpha_{1,2} \geq 0$
- If $f(x, y)$ is convex in x and $p(y) \geq 0 \Rightarrow \int dp(y) f(x, y)$ is convex
- If $f(x, \alpha)$ is convex in $x \Rightarrow g(x) = \max_{\alpha} f(x, \alpha)$ is convex
- If $f(x)$ is convex $\Rightarrow f(Ax + b)$ is convex for any matrix A and vector b
- If $f(x, y)$ is jointly convex in x and $y \Rightarrow g(x) = \min_y f(x, y)$ is convex
- If A and B are convex regions $\Rightarrow A \cap B$ convex region (but same is not true for $A \cup B$ and A^c)
- **Def: convex cone** $S = \{x, y \in S \Rightarrow \alpha x + \beta y \in S \text{ for all } \alpha, \beta \geq 0\}$
- **Def: 2nd order convex cone** $S_2 = \{(x, t) : |x^T x| \leq t^2\}$
- **The only existing convex surface, $h(x) = 0$, is the plane $Ax + b = 0$**

Convex regions:



Convex



Not Convex

Examples of convex functions

- $\exp(x)$
- $-\log(x)$
- x^a
- $a^T x + b$
- $x^T Q x + Lx + r$
- $x \log x$
- $\log \sum_i e^{x_i}$
- $\max_i (a_i^T x + b_i)$
(by above max property)
- $\min_{s \in D} \|x - s\|$
(that is the “distance transform”, by above min property)
- $\max_{s \in D} \|x - s\|$
(by above max property)
- $\sum_i \log(b_i - a_i^T x)^{-1}$
(by above $f(Ax + b)$ and the sum property)

Convex optimization

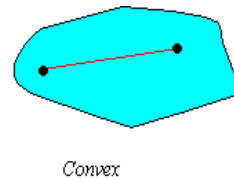
Def: Optimization problem is said to be convex if its objective function is convex and the feasibility region is convex.

$$\min f(x)$$

s.t.

$$g_i(x) \leq 0$$

$$h_i(x) = 0$$



By earlier properties, for optimization problem to be convex $f(x)$ and **all** $g(x)$ must be convex, and **uniquely** $h(x)=Ax+b$

Convex optimization

Convex optimization problem \rightarrow

- $f(x)$ is convex
- all $g_i(x)$ are convex
- $h_i(x) = a_i^T x + b$

Canonical convex optimization problems

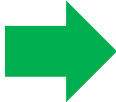
Main convex optimization classes, also known as **canonical convex problems**, all have very efficient solvers implemented for them in software in the past.

If you can reduce your problem to a canonical convex program, good chances are that you can find excellent software that can solve such problem very efficiently and for very large problem sizes.

Canonical convex optimization problems

What you are responsible for: You are responsible for being able to recognize main canonical convex programs and identify the parameters. For example:

	1. LP:	2. Parameters:
$\max_x (x_1 + 2x_2 - 3x_3)$	$\max c^T x$	
$x_1 + 2x_2 \leq 2$	$Cx \leq d$	
$x_1 - 3x_3 \geq 2$		
$x_1 + x_2 + x_3 \leq 3$	$Ax = b$	
$x_1, x_2, x_3 \geq 0$		

	$c = \begin{bmatrix} 1 \\ 2 \\ -3 \end{bmatrix}, C = \begin{bmatrix} 1 & 2 & 0 \\ -1 & 0 & 3 \\ 1 & 1 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, d = \begin{bmatrix} 2 \\ -2 \\ 3 \\ 0 \\ 0 \\ 0 \end{bmatrix}, A = [], b = []$
--	---

Note the sign reversals for 2 and 4,5,6 inequalities.

Canonical convex optimization problems

- Linear Program (LP)

$$\max c^T x$$

$$Cx \leq d$$

$$Ax = b$$

Canonical convex optimization problems

- Quadratic Program (QP)
(Q must be symmetric and positive definite – that is $u^T Q u \geq 0$ for any vector u)

$$\min \frac{1}{2} x^T Q x + c^T x$$

$$C x \leq d$$

$$A x = b$$

Canonical convex optimization problems

- Quadratically Constrained QP (QCQP)
(all Q_i must be symmetric and pos.def.)

$$\min \frac{1}{2} x^T Q_0 x + c_0^T x$$

$$\frac{1}{2} x^T Q_i x + c_i^T x + r_i \leq 0$$

$$Ax = b$$

Canonical convex optimization problems

- Second Order Cone Program (SOCP)

$$\min c_0^T x$$

$$\|Q_i x + b_i\| \leq c_i^T x + d_i$$

$$Ax = b$$

Canonical convex optimization problems

- Generalized linear fractional program (GLFP)

$$\min_x \max_i \frac{c_i^T x + d_i}{e_i^T x + f_i}$$

$$Cx \leq d$$

$$Ax = b$$

$$e_i^T x + f \geq 0$$

Canonical convex optimization problems

- Semi-Definite Programming (SDP)
(positive definiteness constraints on combinations of matrices C and D)

$$\min c^T x$$

$$\sum_k x_k C_k + D \succ 0$$

$$Ax = b$$



$$\min \text{Tr}(C^T X)$$

$$\text{Tr}(A_i^T X) = b_i$$

$$X \succ 0$$

Canonical convex optimization problems

- Determinant maximization program (MAXDET)

$$\min \ln \det \left[\sum_k x_k G_k + F \right] - c^T x$$

$$\sum_k x_k G_k + F \succ 0$$

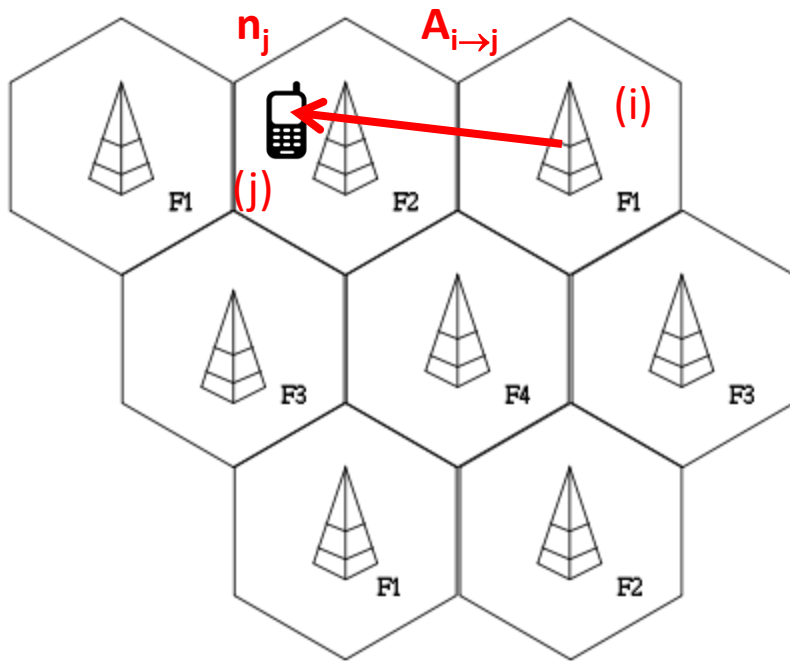
$$\sum_k x_k C_k + D \succ 0$$

$$Ax = b$$

Canonical convex optimization problems

You have to: Be able to recognize and correctly formulate parameters for LP, QP, QCQP, SOCP, and GLFP type of convex problems.

Example: optimal cell-tower power allocation



Signal strength at (j):

$$A_{i \rightarrow j} P_i$$

Signal quality at (j):

$$\frac{A_{i \rightarrow j} P_i}{\sum_{i' \neq i} A_{i' \rightarrow j} P_{i'} + n_j}$$

GLFP

$$\min_x \max_i \frac{c_i^T x + d_i}{e_i^T x + f_i}$$

$$Cx \leq d$$

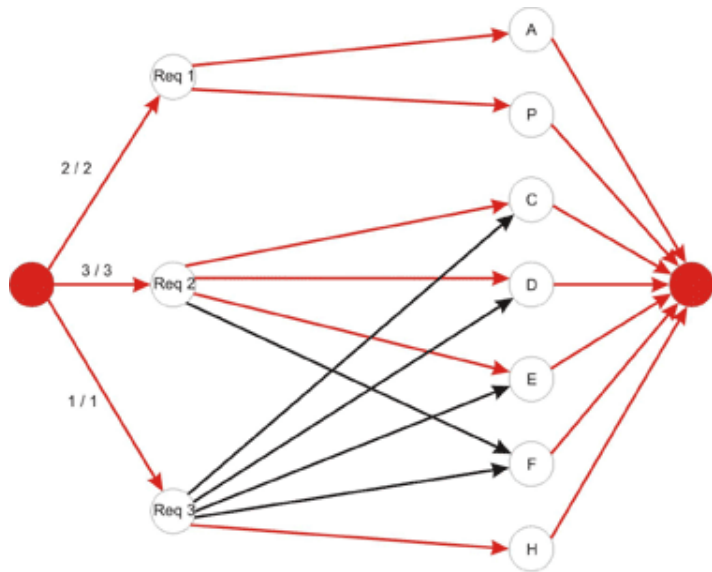
$$Ax = b$$

$$e_i^T x + f_i \geq 0$$

Power allocation:

$$\max_P \min_{ij} \frac{A_{i \rightarrow j} P_i}{\sum_{i' \neq i} A_{i' \rightarrow j} P_{i'} + n_j}, 0 \leq P_i \leq P_{\max}$$

Example: maximal flow problem



$$\max \sum_j F_{\text{source} \rightarrow j}$$

$$F_{i \rightarrow j} \leq F_{\max; i \rightarrow j}$$

$$F_{i \rightarrow j} \geq 0$$

$$\sum_i F_{i \rightarrow j} = \sum_i F_{j \rightarrow i}, \text{ all } j \neq \{\text{source}, \text{sink}\}$$

LP

$$\min c^T x$$

$$Cx \leq d$$

$$Ax = b$$

Issues in large scale optimization

Large scale optimization may refer to solving problems either with a large number of parameters ($p=1000$ or more) or/and large sets of data ($n=1,000,000$ or more)

Two main issues in LSOpt:

- The computational cost of linear algebra operations in numerical optimization algorithms such as H^{-1} – $O(p^3)$
- The computational cost of evaluating the objective function f and/or its derivatives $\nabla f, Hf$

Issues in large scale optimization

With respect to the 1st problem, different optimization methods show dramatically different ability to scale towards large p .

GD typically is one of the least expensive numerical optimization method to use and requires only *the calculation of the gradients* followed by an easy 1D line-search optimization, thus having the least complexity of all numerical optimization algorithms - $O(p)$.

Issues in large scale optimization

NM demands $O(p^3)$ time complexity for Hessian inversion and QNM is $O(p^2)$ time complexity due to matrix multiplications.

For NM, a possible strategy in large scale situation is to replace the Hessian inversion with an approximate solution of the step-equation $H\Delta x = -g$ instead, via a Conjugate Gradient-like method.

Issues in large scale optimization

CGM is a method for approximately solving systems of linear equations $Ax=b$ whereas A is symmetric - exactly the situation with the Hessian matrix, which is always symmetric.

CGM is exact if run for p steps, but also can be used as an approximate linear solver if stopped early – which is exactly what we want to do here.

Issues in large scale optimization

CGM was reviewed in the Advanced part of the lecture on vectors and matrices. For here, it will be sufficient to say that with CGM the time complexity of NM algorithm's step, $H\Delta x = -g$, reduces to $O(\kappa(H)p^2)$, where $\kappa(H)$ is the condition number of the Hessian matrix H , and becomes comparable to quasi-Newton method.

Issues in large scale optimization

With respect to the 2nd problem – large dataset
– all numerical optimization methods suffer in equal degree.

For example, consider Levenberg-Marquardt optimization of a RSS loss function:

$$S(\theta) = \frac{1}{2} \sum_{i=1}^n (y_i - f(x_i; \theta))^2$$

$$\left[\sum_{i=1}^n g_i g_i^T + \lambda \text{diag} \left(\sum_{i=1}^n g_i g_i^T \right) \right] \Delta \theta = \sum_{i=1}^n g_i \cdot (y_i - f(x_i; \theta))$$

Issues in large scale optimization

With $n=1,000,000$ examples (x_i, y_i) in dataset, each iteration of LM requires summing up 1,000,000 values just to evaluate the gradient and the Hessian !

This situation is not much better for GD algorithm:

$$S(\theta) = \frac{1}{2} \sum_{i=1}^n (y_i - f(x_i; \theta))^2$$

$$\nabla S(\theta) = \sum_{i=1}^n g_i \cdot (f(x_i; \theta) - y_i)$$

Issues in large scale optimization

Large dataset may influence dramatically the possibility of constructing optimal ML solutions.

The solutions currently employed are **batch** and **stochastic** optimization.

Issues in large scale optimization

In **batch optimization**, the full dataset is broken into batches of, for example, $n_b=1000$ data points.

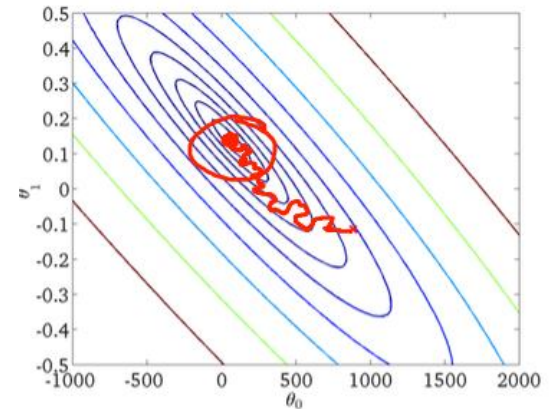
A numerical optimization algorithm is then used with that small batches of data to approach the optimal solution in some way.

Issues in large scale optimization

(mini)Batch Gradient Descent

$$\theta = \theta - \alpha \nabla_{\theta} S(\theta; x_{k:k+nb}; y_{k:k+nb})$$

$$S(\theta; x_{k:k+nb}; y_{k:k+nb}) = \frac{1}{2} \sum_{i=k}^{k+nb} (y_i - f(x_i; \theta))^2$$



There is much leeway currently in how exactly this process is done; batches may be reused circularly until convergence, batches may be selected randomly, learning rate may decrease as more batches are processed, etc.

Issues in large scale optimization

Stochastic Gradient Descent (SGD) is basically the minibatch optimization with $nb=1$

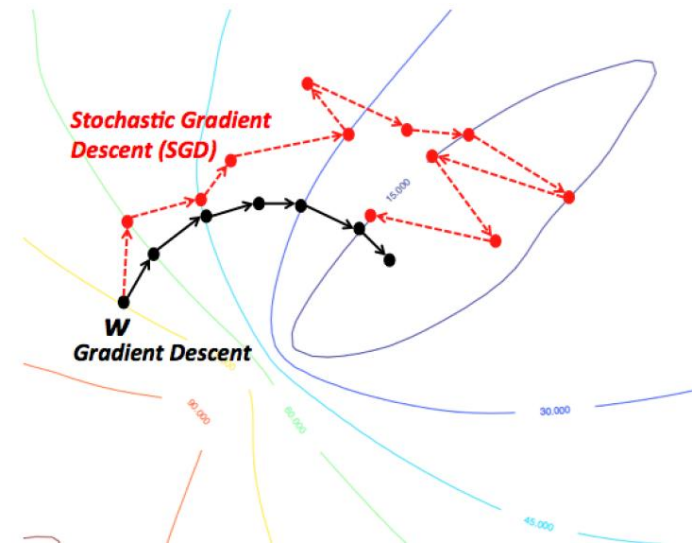
- For each next step of GD, SGD uses a single data point to evaluate the gradient direction
- Convergence is then expected “on average”, with steps away and towards the optimal solution *averaging-out* once SGD moves close enough to a solution

Issues in large scale optimization

Stochastic Gradient Descent

$$\theta = \theta - \alpha \nabla_{\theta} S(\theta; x_k; y_k)$$

$$S(\theta; x_k; y_k) = \frac{1}{2} (y_k - f(x_k; \theta))^2$$



SGD is currently the method of choice for training large CNN and deep neural networks (DNN)

Note: The order of examples may need to be randomized to avoid silly problems with non-random data inputs.

Issues in large scale optimization

Lastly, where computational cost cannot be reduced further, parallelizing computation over clusters or supercomputing infrastructures takes full effect.

In that regard, Google's **MapReduce** has been one of the key players in the industry in the past years (we will not touch this topic here).

QUESTIONS FOR SELF-CONTROL

- Define what is meant by the standard form of optimization problems
- Solve manually using the KKT conditions $\min(2x+y)$ s.t. $x^2+y^2 \leq 1$
- Explain how log-barrier constrained numerical optimization method works to solve constrained optimization problems
- Write down the objective function modified by the log-barrier, $L_t(x)$, for the optimization problem in 2nd question
- Inspect manually the log-barrier method for a simple optimization problem $\min x^2$ s.t. $x > 1$. For that: write down $L_t(x)$ and analytically find the minimum from the condition $dL_t(x)/dx=0$. How does that minimum change as $t \rightarrow 0$?
- Modify the previous question by making the constraint $x > -1$. How does the log-barrier'ed minimum behave with t now?
- Define the dual function of an optimization problem.
- What is primal problem? What is feasible region and feasible point?
- Define dual optimization problem. How does dual problem allow one to bound the solution of the primal problem?
- Define strong duality.
- Define convex function and convex region.
- Define convex optimization problem.

- Define the standard form of convex optimization problems.
- Name the main (canonical) classes of convex optimization problems – the convex programs.
- Write down some simple examples of LP, QP, QCQP and GLFP.
- Identify the canonical problem type and write down the corresponding parameters for the following convex problem: $\min x^2 - 2xy + 2y^2 - 2x + y$ s.t. $x + 2y \leq 3, y \geq 0$
- How is the maximal flow problem a Linear Program? Construct the LP parameter matrices for the max-flow problem.
- What is mean by the large-scale optimization?
- What are the two main problems faced in large-scale optimization?
- What is time complexity of Gradient Descent with respect to the problem size p and dataset size n ?
- What is time complexity of Newton Method with respect to the problem size p and dataset size n ?
- What is time complexity of quasi-Newton Method with respect to the problem size p and dataset size n ?
- Explain the main idea behind using Conjugate-Gradient in large scale Newton optimization method.

- Think about and argue when to use each of the 3 main numerical optimization algorithms we discussed – GD, NM, QNM. For that think and write in a table format the advantages and disadvantages of the three.
- Describe the main idea behind the mini-batch and stochastic GD methods.
- How would you parallelize the optimization of a loss function of a large ML neural network having $p=100,000$ parameters and $n=10,000,000$ training data, optimized by stochastic gradient descent method on a cluster of computers?
- **Advanced:** explain how Conjugate Gradient method works.

ADVANCED

Duality theory

The function

$$G(\lambda, \mu) = \min_{any\ x} L(x, \lambda, \mu) = \min_{any\ x} \left[f(x) + \sum_{i=1}^q \lambda_i h_i(x) + \sum_{i=1}^r \mu_i g_i(x) \right]$$

is called **dual function** for an optimization problem $(\min f, g_i \leq 0, h_i = 0)$.

*That is, the dual function is a **function of Lagrange multipliers** in which x is (unconstrained) optimized over*

Duality theory

Some interesting properties of dual function:

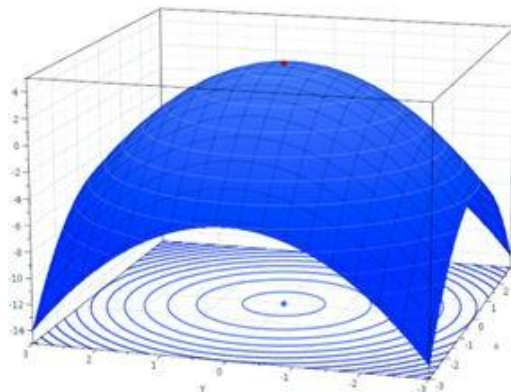
- **THM1:** If $f^* = \{ \min f(x) \text{ s.t. } g_i(x) \leq 0 \text{ and } h_i(x) = 0 \}$, then $G(\lambda, \mu) \leq f^*$ for all $\mu \geq 0$.
- **THM2 (Duality Property):** If $G^* = \{ \max g(\lambda, \mu) \text{ s.t. } \mu \geq 0 \}$, then $G^* \leq f^*$.

That is, $g(\lambda, \mu)$ can be used to bound the solution of the original problem f^* from below –a very valuable property.

Duality theory

- **THM3:** $g(\lambda, \mu)$ is always concave.

That is, $g(\lambda, \mu)$ is easy to maximize numerically – it contains no local maxima, one can just follow the gradient all the way up and find the global maximum.



← Concave function

Duality theory

In optimization, the problem $\min f(x)$ s.t. $g_i(x) \leq 0$, $h_i(x) = 0$ is called **the primal problem** and the problem $\max g(\lambda, \mu)$ s.t. $\mu \geq 0$ is called **the dual problem**

Note: if we start with a problem **$\max f$** then we would end up with the dual problem **$\min g$** , recall that $\max f = \min -f$, and the duality theorems then obviously is all reversed (**TRY TO CHECK THIS AT HOME**)

Duality theory

- **Strong duality** holds if $G^* = f^*$.
- In strong duality, the primal problem can be solved exactly through the dual problem !
- *Strong duality* is generally related to the geometry of the constraints in an optimization problem, and normally holds for sufficiently “nice” problems.

Duality theory

- Strong duality can be used to solve some optimization problems more simply, by converting them to their dual form.
- Even if strong duality is not known to hold, the condition $G^*=f^*$ can be verified after the solution, which is still easier to do in many cases than solving the original primal problem.

Duality theory

Example: constrained linear optimization (note max here instead of min and remember the reduction to standard form and conversion between max and min)

$$L(x, \lambda, \mu, \nu) = c^T x - \lambda^T (Bx - d) - \mu^T (Ax - b) + \nu^T x$$

$$\max_x c^T x$$

s.t.

$$Ax \leq b$$

$$Bx = d$$

$$x \geq 0$$



$$G(\lambda, \mu, \nu) = d^T \lambda + b^T \mu$$

$$\text{s.t. } -c + B^T \lambda + A^T \mu - \nu = 0$$

Duality theory

The dual problem obtained here is arguably simpler because the inequality constraint has been converted to μ, ν -quadrant restriction, which is much a simpler geometric constraint.

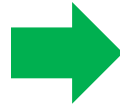
$$\max_x c^T x$$

s.t.

$$Ax \leq b$$

$$Bx = d$$

$$x \geq 0$$



$$\min_{\lambda, \mu} (d^T \lambda + b^T \mu)$$

s.t.

$$B^T \lambda + A^T \mu - \nu = c$$

$$\mu, \nu \geq 0$$

Duality theory

Dual problem bounds solution from above:

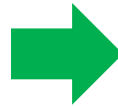
$$\max_x c^T x$$

s.t.

$$Ax \leq b$$

$$Bx = d$$

$$x \geq 0$$



$$\min_{\lambda, \mu} [d^T \lambda + b^T \mu]$$

s.t.

$$A^T \mu + B^T \lambda \geq c$$

$$\mu \geq 0$$

Try feasible $x_1=2 \Rightarrow f^* \geq 2$

$$\max_x (x_1 + 2x_2 - 3x_3)$$

$$x_1 + 2x_2 \leq 2$$

$$x_1 - 3x_3 \geq 2$$

$$x_1 + x_2 + x_3 \leq 3$$

$$x_1, x_2, x_3 \geq 0$$



$$f^* \geq 2 \text{ \& } G^* \leq 2 \text{ \& } G^* \geq f^* \\ \Rightarrow f^* = G^* = 2$$

Try feasible $\mu_1=1 \Rightarrow G^* \leq 2$

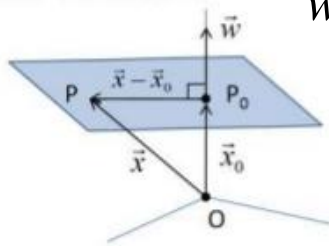
$$\min_{\mu} (2\mu_1 - 2\mu_2 + 2\mu_3)$$

$$\begin{bmatrix} 1 & -1 & 1 & -1 & 0 & 0 \\ 2 & 0 & 1 & 0 & -1 & 0 \\ 0 & 3 & 1 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix} \geq \begin{bmatrix} 1 \\ 2 \\ -3 \end{bmatrix}$$

$$\mu_i \geq 0$$

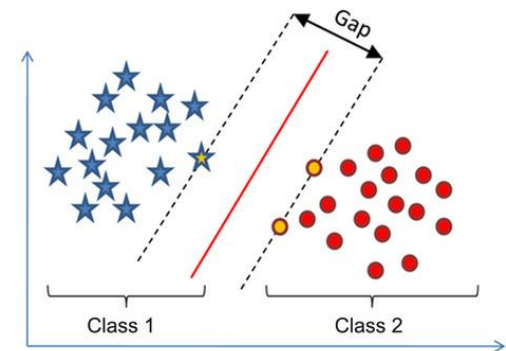
Duality theory

Support Vector Machine – a maximal margin classifier:



$$w^T \cdot (x - P_0) = w^T x + b = 0$$

An equation of a hyperplane is defined by a point (P₀) and a perpendicular vector to the plane (\vec{w}) at that point.



Maximal margin linear classifier

Duality theory

The class separation condition in SVM is written as $y_i(w^T x_i + b) \geq 1$, for all i .

If $y_i(w^T x_i + b) \geq 1$, the geometric margin between classes can be shown to be $1/|w|$. Thus, maximizing margin is:

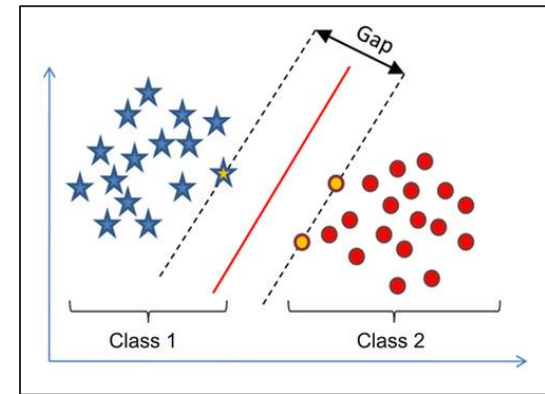
$$\max 1/|w|$$

$$s.t. y_i(w^T x_i + b) \geq 1 \text{ all } i$$



$$\min |w|^2$$

$$s.t. y_i(w^T x_i + b) \geq 1 \text{ all } i$$



Duality theory

This is a quadratic optimization problem. We write the Lagrange function for it as follows:

$$L(w, b, \alpha) = \frac{1}{2} w^T w - \sum \alpha_i (y_i (w^T x_i + b) - 1)$$

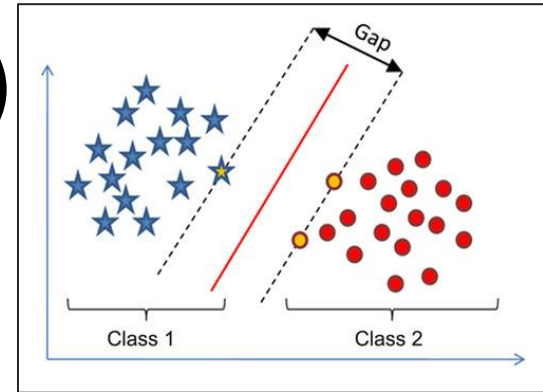
Unconstrained optimization in (w, b)

$$\partial_b \rightarrow \sum \alpha_i y_i = 0$$

$$\partial_w \rightarrow w = \sum \alpha_i y_i x_i$$

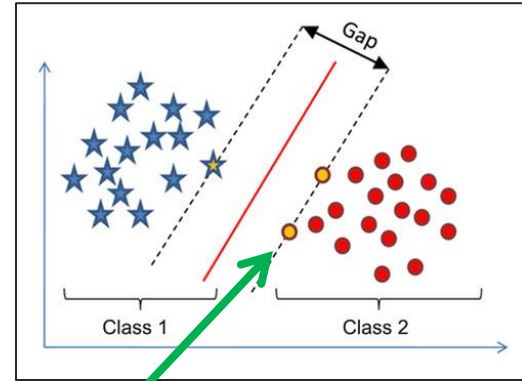
Dual function $G(\alpha)$ is

$$G(\alpha) = \sum \alpha_i - \frac{1}{2} \sum \alpha_i \alpha_j y_i y_j (x_i^T \cdot x_j)$$



Duality theory

The dual problem here is simpler than the original. In addition, the w-optimality condition indicates that the solution is determined by a small number of **support vectors**:



$$\partial_w \rightarrow w = \sum \alpha_i y_i x_i$$

Solution is determined by a small number of **support vectors** !

What's more, we discover that only the **dot-products of the feature vectors** x_i are actually required to find the solution – the situation known in ML as the kernel property – the original features are actually quite irrelevant.

$$\max \left(\sum \alpha_i - \frac{1}{2} \sum \alpha_i \alpha_j y_i y_j (x_i^T \cdot x_j) \right)$$
$$s.t. \alpha_i \geq 0, \sum \alpha_i y_i = 0$$

Only the dot-products are needed for solution, original features are irrelevant

Primal-dual method

- Primal-dual interior point method (PDM) is a more rigorous method for solving general constrained optimization problems.
- It solves the primal and the dual problems in the KKT equations simultaneously, by making Newton steps that attempt to minimize the objective function while also keeping all constraints true.

Primal-dual method

PDM solves the KKT equations directly in primal-dual space, while also replacing zeros in the RHS of the complementary slackness condition with a small negative constant t that is supposed to gradually decrease to zero:

$$\nabla f(x) + \sum_{i=1}^q \lambda_i \nabla h_i(x) + \sum_{i=1}^r \mu_i \nabla g_i(x) = 0$$

$$g_i(x) \leq 0$$

$$h_i(x) = 0$$

$$\mu_i \geq 0$$

$$\mu_i g_i(x) = -t, t > 0, t \rightarrow 0$$

Primal-dual method

The solution of KKT equations for last t_k is used as the starting point for the next iteration t_{k+1} while t_k is reduced by a constant factor γ . If the final solution is in the interior, it is found. If the solution is on the boundary ($g_i(x)=0$), it is recovered as $t_k \rightarrow 0$.

$$\nabla f(x) + \sum_{i=1}^q \lambda_i \nabla h_i(x) + \sum_{i=1}^r \mu_i \nabla g_i(x) = 0$$

$$g_i(x) \leq 0$$

$$h_i(x) = 0$$

$$\mu_i \geq 0$$

$$\mu_i g_i(x) = -t_k, t_{k+1} = \gamma \cdot t_k, \gamma < 1$$

Primal-dual method

Note that the slackness conditions can be resolved as $g_i(x) = -t/\mu_i$, which together with $\mu_i \geq 0$ automatically implies $g_i(x) < 0$. With that, $g_i(x) \leq 0$ constraints can be entirely removed from the problem in favor of simpler constraints $\mu_i \geq 0$:

$$\nabla f(x) + \sum_{i=1}^q \lambda_i \nabla h_i(x) + \sum_{i=1}^r \mu_i \nabla g_i(x) = 0$$

~~$$g_i(x) \leq 0$$~~

$$h_i(x) = 0$$

$$\mu_i \geq 0$$

$$\mu_i g_i(x) = -t_k \Rightarrow g_i(x) = -t_k / \mu_i < 0$$

Primal-dual method

The remaining equations have only equality constraints together with a simple domain-positivity $\mu \geq 0$:

$$\nabla f(x) + \sum_{i=1}^q \lambda_i \nabla h_i(x) + \sum_{i=1}^r \mu_i \nabla g_i(x) = 0$$

$$\mu_i g_i(x) = -t_k$$

$$h_i(x) = 0$$

$$\mu_i \geq 0$$

Primal-dual method

These reduced KKT equations are solved using the Newton method for **constrained optimal point** with **backtracking**.

Primal-dual method

For the discussion of this modified Newton method, we will discard the h-constraints for simplicity (this can be straightforwardly fixed) and imagine that we work with the following problem:

$$\nabla f(x) + \sum_{i=1}^r \mu_i \nabla g_i(x) = 0$$

$$\mu_i g_i(x) = -t$$

$$\mu_i \geq 0$$

Primal-dual method

The modified Newton method makes steps in the **primal-dual space** (x, μ) in the direction guided by **the Hessian of the complete Lagrange function** $L(x, \mu)$:

$$Hf(x)\Delta x + \sum \mu_i Hg_i(x)\Delta x + \sum \nabla g_i(x)\Delta \mu_i = -(\nabla f(x) + \sum \mu_i \nabla g_i(x))$$

which can be re-written as

$$H_\mu(x)\Delta x + A(x)\Delta \mu = -\nabla f(x) - A(x)\mu$$

where

$$H_\mu(x) = Hf(x) + \sum \mu_i Hg_i(x), A(x) = [\nabla g_1(x), \nabla g_2(x), \dots, \nabla g_r(x)]$$

(H_μ is the Hessian of the objective function **plus** the constraints; A is the gradients of g_i stacked as columns into a rectangular matrix).

Primal-dual method

The full Hessian matrix of the complete Lagrange function BTW is:

$$HL(x, \mu) = \begin{bmatrix} H_{\mu}(x) & A(x) \\ A(x)^T & 0 \end{bmatrix}$$

Primal-dual method

It is further requested that the constraint $g(x)\mu = -t$ stays satisfied during the Newton step:

- If the constraint was satisfied before the step (*feasible starting point*), then this requires

$$\mu_i \nabla g_i(x)^T \Delta x + g_i(x) \Delta \mu_i = 0$$

- If the constraint was not satisfied at the start (*infeasible starting point*, $g(x)\mu + t \neq 0$), then this implies

$$\mu_i \nabla g_i(x)^T \Delta x + g_i(x) \Delta \mu_i = -t - g_i(x)\mu_i$$

Primal-dual method

We continue with the second condition, the infeasible starting point. Then, the the Newton optimality condition plus the constraint lead to the following system of linear equations for Δx and $\Delta \mu$:

$$H_{\mu}(x)\Delta x + A(x)\Delta \mu = -\nabla f(x) - A(x)\mu$$

$$\Lambda(x)\Delta x + G(x)\Delta \mu = -t \cdot \vec{1} - G(x)\vec{\mu}$$

Primal-dual method

In the matrix notation, this is:

$$\begin{bmatrix} H_{\mu_k}(x_k) & A(x_k) \\ \Lambda(x_k) & G(x_k) \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta \mu_k \end{bmatrix} = \begin{bmatrix} -\nabla f(x_k) - A(x_k)\mu_k \\ -t \cdot \vec{1} - G(x_k)\mu_k \end{bmatrix}$$

$$A(x) = [\nabla g_1(x), \nabla g_2(x), \dots]$$

$$\Lambda(x) = \begin{bmatrix} \mu_1 \nabla g_1(x)^T \\ \mu_2 \nabla g_2(x)^T \\ \dots \end{bmatrix} \quad G(x) = \begin{bmatrix} g_1(x) & 0 & \dots \\ 0 & g_2(x) & \dots \\ \dots & \dots & \dots \end{bmatrix}$$

Primal-dual method

This Newton step can be implemented as usual, by inverting the system's matrix. After the step, the objective function $f(x)$ is improved as is the constraint's feasibility, $g(x)\mu + t = 0$

$$(x_k, \mu_k) \rightarrow (x_{k+1} = x_k + \Delta x_k, \mu_{k+1} = \mu_k + \Delta \mu_k)$$

$$\nabla f(x_k) + A(x_k)\mu_k \rightarrow 0$$

$$t \cdot \vec{1} + G(x_k)\mu_k \rightarrow 0$$

Primal-dual method

Backtracking: In the above Newton step, the domain positivity constraint $\mu \geq 0$ may become violated as it is not controlled for:

$$\mu_k \rightarrow \mu_{k+1} = \mu_k + \Delta\mu_k < 0$$

If that happens, the method **backtracks** – that is, it shrinks the step back towards (x_k, μ_k) , while keeping the same direction $(x_k, \mu_k) \rightarrow (x_{k+1}, \mu_{k+1})$ until μ -positivity is restored.

The relationship between the primal-dual and the log-barrier method

$$\left\{ \begin{array}{l} \nabla f(x) + \sum_{i=1}^r \mu_i \nabla g_i(x) = 0 \\ \mu_i g_i(x) = -t \Rightarrow \mu_i = -\frac{t}{g_i(x)} \end{array} \right. \Rightarrow$$

$$\Rightarrow \nabla f(x) - t \sum_{i=1}^r \frac{\nabla g_i(x)}{g_i(x)} = \nabla \left(f(x) - t \sum_{i=1}^r \ln(-g_i(x)) \right) = \nabla L_t(x) = 0$$