Weighted Least-Squares

- Least-squares regression problem:
  - Basis functions: \( f_1(x), f_2(x), \ldots, f_n(x) \)
  - Find coefficients: \( w_1, \ldots, w_k \)
  \[
  \min_{w} \sum_j \left( t_j - \sum_i w_i f_i(x_j) \right)^2
  \]

- Some points are more important than others:
  - Weighted least-squares:
  \[
  \min_{w} \sum_j \alpha_j \left( t_j - \sum_i w_i f_i(x_j) \right)^2
  \]
  if care more about \( j \) than \( \alpha_j \) is larger...
Robust Least Squares

- Weighted least squares:
  - Test set distribution may be different from training set!
    - Must reweigh according to likelihood ratio:
      \[ \alpha_j = \frac{\epsilon_j(x_j)}{P(x_j)} \]

- But what is the test set distribution???
- Don’t want to commit!
  - Pick worst case weights!
  - Robust LS:
    \[
    \max \alpha \min \sum_j \alpha_j (t_j - \sum_i w_i f_i(x_j))^2 \\
    \alpha > 0 \quad \sum \alpha_j = 1
    \]

Optimization of Robust LS

- Robust LS problem:
  - For each set of weights, must solve weighted least squares:
  - How do we find worst case weights?
    - Option B : guess weights, solve least squares, tweak weights,…
Equivalent optimization problem

- Robust LS:
  - Pushing min $w$ into constraint:
    - Non-linear constraint, give up!

Minimum over $w$ as infinite constraints

- Non-linear min constraint:
  - Infinite constraint set:
    - Great! Had a non-linear constraint, now all I have are infinite constraints, for each $w$!
Constraints for one alpha, help with other alphas

- Suppose you have $\alpha^0$, and introduce a constraint for some coefficients $w_0$:
  - Constraint also upper bound for other weights $\alpha$:
    - Linear constraint! Cool!

A geometric view

- We have an infinite number of linear constraints, many are irrelevant
  - Set of constraints forms a convex set
    - Linear program with one constraint per $w$
      - Still infinite…
Suppose we use a subset of the constraints

- What if we use a finite number of constraints
  - Set of constraints at a finite set of coefficients $\Omega$

- Can solve with any LP solver!
- But, solution with subset of constraints may not be a solution to original problem
  - Fewer constraints, solution may be infeasible, value of LP too high…

Active constraints

- Original LP with infinite constraints:
  - How many variables?
  - How many active constraints at optimal solution?

- So, if we knew set of active constraints at optimal solution $\Omega^*$
  - Could discard all other constraints
Active Constraints at Optimal Point

- Original problem:
  - If we knew set of active constraints at optimal solution $\Omega^*$
    - Could discard all other constraints
    - Solution will be feasible with respect to original problem

- Consider some set of constraints $\Omega$:
  - Too few, infeasible solution:
  - Just right, feasible solution:

Constraint Generation

- Start with some finite set of constraints $\Omega$
  - Solve LP, obtain $\alpha$

- Check is $(c_0, \alpha_0)$ is feasible for infinite constraints:
  - If feasible, done!
  - Otherwise, add a constraint that makes $(c_0, \alpha_0)$ infeasible:

- But how do we find which constraint to add???
Separation Oracle for Robust LS

Original problem:

- Is \((\varepsilon_0, \alpha_0)\) feasible?
  - infeasibility \(\varepsilon_0\) too high for this particular \(\alpha_0\)

- What’s the smallest possible \(\varepsilon\)?

- Standard weighted LS!
  - If result is \(\varepsilon\), then we are done!
  - Otherwise found a violated constraint

Are we there yet?

- When do we stop?
  - Solve with infinite set of constraints:
    - Obtain \((\varepsilon_{opt}, \alpha_{opt})\)

  - Solve with constraints \(\Omega\)
    - Obtain \((\varepsilon_0, \alpha_0)\)

  - Optimizing subset of constraints, same objective

  - If we get any feasible point with infinite constraints
    - E.g.,

  - Bound on how far we are from optimal solution:
Constraint Generation: 
The General Case

- Given an LP with (possibly infinitely) many constraints:
  - Start with some subset of the constraints
  - Solve LP to find a solution with new subset of the constraints:
    - Separation oracle:
      - If \( x_\Omega \) is feasible:
      - If \( x_\Omega \) is infeasible:
    - Add violated constraint to set
    - (It is also possible to remove (some or all) inactive constraints, in addition to adding violated constraints)
      - Makes LP solver step faster
      - But requires more outer loop iterations
      - Trade-off is application specific

Bound on optimal solution - General case

- Problem with many constraints:
  - Some relaxation:
    - E.g., only subset of constraints
  - If you can obtain some feasible point for the original problem:
  - Bound on the optimal solution:
Why constraint generation converges

- LP with many many constraints:
  - Solve with subset of constraints:
    - (also called “cutting planes”)
  - Relaxed problem, bound on objective:
    - If solution $x_\Omega$ is feasible wrt all constraints:
    - If solution $x_\Omega$ is infeasible wrt all constraints:

Practicalities of Constraint Generation

- Constraint generation converges in a finite number of iterations if the original set is finite
  - Can’t guarantee fast rate in general, similar to simplex algorithm (there are special cases with good rates)
  - Infinite case: will get arbitrarily close, but not necessarily to the optimum
- Idea of using relaxations to obtain bounds is very useful in general
  - E.g., useful in duality (more later in the semester)
- Separation oracle:
  - Must find some violated constraint
  - If we find most violated constraint, usually faster
  - Also very useful for proving that LPs can be solved in polytime (ellipsoid algorithm, more later)
- Constraint generation is extremely useful in practice
  - Often, e.g., robust LS, we have a poly-time separation oracle, even if there are exponentially or infinitely many constraints
  - Even if polynomially many constraints, a fast oracle can make constraint generation faster than using a standard solver
- Constraint generation can be useful for solving general convex problems, not just LP
- Remember: most LP solvers allow you to start from previous solution
  - (the one found with fewer constraints)
  - Make sure you do this, otherwise approach will be much much much slower
Constraint generation and duality

- Primal problem with many constraints:
  \[
  \max_x \sum_j b_j x_j \\
  s.t. \quad \sum_j a_{ij} x_j \leq c_i, \quad \forall i \in I
  \]
  - Constraint generation: find most important constraints
  - What’s the dual equivalent?

- Dual:
Column generation
(aka variable generation)

- Dual problem:
  \[ \min_y \sum_{i \in I} c_i y_i \]
  \[ s.t. \] \[ \sum_{i \in I} a_{ij} y_i = b_j, \quad \forall j \in 1, \ldots, m \]
  \[ y_i \geq 0, \quad \forall i \in I \]

  - Many many variables!!
  - At optimal basic feasible solution
    - Most variables are zero

- Idea:
  - Set most variables to zero
  - Solve problem with other variables:
    - Incrementally increase sets of non-zero variables

Solving problem with subset of variables

- Solve problem with subset of variables
  \[ \min_y \sum_{i \in \Omega} c_i y_i \]
  \[ s.t. \] \[ \sum_{i \in \Omega} a_{ij} y_i = b_j, \quad \forall j \in 1, \ldots, m \]
  \[ y_i \geq 0, \quad \forall i \in \Omega \]

- Rest of variables set to zero

- Questions:
  - How do we decide what variables to use?
  - How do we decide when we are done?
What variables should we add?

- Same as simplex
- Solve problem with variables $\Omega$
  - At optimal basic feasible solution, set of basic variables $B$
- Find submatrix corresponding to basic variables $A_B$
  - Cost of these variables $c_B$
- Reduced cost for each potential new variable $y_i$, for $i \in I$:
  - If all are positive?
  - Otherwise:
- Guaranteed to converge to optimal solution

Column generation summary

- Dual of constraint generation
- Also useful for problems with infinitely many variables
- Some problems
  - Have efficient separation oracles
    - In these, constraint generation is useful
  - Have efficient variable generation oracles
    - In these, column generation is useful
- Both methods can be useful in polynomially large problems
  - E.g., when constraint matrix is too large to fit in memory
    - By incrementally solving the problem, bound amount of memory needed at each iteration
- If you have many many variables and constraints
  - Can use a combination of constraint and column generation