OPTIONAL: Primal Active-Set Algorithm (Lazy Projection Methods)

Recall that Projected Gradient Descent tried to satisfy all the constraints in the projection step

Quadratic Optimization: Primal Active-Set Algorithm

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} + \beta \\ \text{subject to} & A\mathbf{x} \geq \mathbf{b} \end{array} \tag{88}$$

where $Q \succ 0$. The KKT conditions are:

Quadratic Optimization: Primal Active-Set Algorithm

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} + \beta \\ \text{subject to} & A\mathbf{x} \geq \mathbf{b} \end{array} \tag{88}$$

where $Q \succ 0$. The KKT conditions are:

- $Q\widehat{\mathbf{x}} + c - \sum_{i=1}^{m} \widehat{\lambda}_i \mathbf{a}_i = 0$
- $\widehat{\lambda}_i(\mathbf{a}_i^T\widehat{\mathbf{x}} - b_i) = 0$ for $i = 1..m$
- $\widehat{\lambda}_i \geq 0$ for $i = 1..m$
- $A\widehat{\mathbf{x}} \geq \mathbf{b}$...

Quadratic Optimization: Primal Active-Set Algorithm

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} + \beta \\ \text{subject to} & A\mathbf{x} \geq \mathbf{b} \end{array} \tag{88}$$

where $Q \succ 0$. The KKT conditions are:

- $Q\widehat{\mathbf{x}} + c - \sum_{i=1}^{m} \widehat{\lambda}_i \mathbf{a}_i = 0$
- $\widehat{\lambda}_i(\mathbf{a}_i^T\widehat{\mathbf{x}} - b_i) = 0$ for $i = 1..m$
- $\widehat{\lambda}_i \geq 0$ for $i = 1..m$
- $A\widehat{\mathbf{x}} \geq \mathbf{b}$... If $\widehat{\mathbf{x}}$ lies in interior of feasible region then

Quadratic Optimization: Primal Active-Set Algorithm

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} + \beta \\ \text{subject to} & A\mathbf{x} \geq \mathbf{b} \end{array} \tag{88}$$

where $Q \succ 0$. The KKT conditions are:

- $Q\widehat{\mathbf{x}} + c - \sum_{i=1}^{m} \widehat{\lambda}_i \mathbf{a}_i = 0$
- $\widehat{\lambda}_i(\mathbf{a}_i^T\widehat{\mathbf{x}} - b_i) = 0$ for $i = 1..m$
- $\widehat{\lambda}_i \geq 0$ for $i = 1..m$
- $A\widehat{\mathbf{x}} \geq \mathbf{b}$... If $\widehat{\mathbf{x}}$ lies in interior of feasible region then
  1. $\widehat{\lambda} = 0$
  2. $\widehat{\mathbf{x}} = -Q^{-1}\mathbf{c}$

Quadratic Optimization: Primal Active-Set Algorithm

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} + \beta \\ \text{subject to} & A\mathbf{x} \geq \mathbf{b} \end{array} \tag{89}$$

where $Q \succ 0$. The KKT conditions are:

- $Q\widehat{\mathbf{x}} + c - \sum_{i=1}^{m} \widehat{\lambda}_i \mathbf{a}_i = 0$
- $\widehat{\lambda}_i(\mathbf{a}_i^T\widehat{\mathbf{x}} - b_i) = 0$ for $i = 1..m$
- $\widehat{\lambda}_i \geq 0$ for $i = 1..m$
- $A\widehat{\mathbf{x}} \geq \mathbf{b}$... If some $\underline{\mathbf{a}_i^T\mathbf{x}^* = b_i}$ for some $i \in I^*$ (index set of active constraints) then

Quadratic Optimization: Primal Active-Set Algorithm

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q \mathbf{x} + \mathbf{c}^T \mathbf{x} + \beta \\ \text{subject to} \quad & A\mathbf{x} \geq \mathbf{b} \end{aligned} \tag{89}$$

where $Q \succ 0$. The KKT conditions are:

- $Q\widehat{\mathbf{x}} + c - \sum_{i=1}^{m} \widehat{\lambda}_i \mathbf{a}_i = 0$
- $\widehat{\lambda}_i(\mathbf{a}_i^T \widehat{\mathbf{x}} - b_i) = 0$ for $i = 1..m$
- $\widehat{\lambda}_i \geq 0$ for $i = 1..m$
- $A\widehat{\mathbf{x}} \geq \mathbf{b}$... If some $\mathbf{a}_i^T \mathbf{x}^* = b_i$ for some $i \in I^*$ (index set of active constraints) then , one needs to iteratively solve $\mathbf{x}^k$ and I_k

Quadratic Optimization: Primal Active-Set Algorithm

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} + \beta \\ \text{subject to} & A\mathbf{x} \geq \mathbf{b} \end{array} \tag{89}$$

where $Q \succ 0$. The KKT conditions are:

- $Q\widehat{\mathbf{x}} + c - \sum_{i=1}^{m} \widehat{\lambda}_i \mathbf{a}_i = 0$
- $\widehat{\lambda}_i(\mathbf{a}_i^T\widehat{\mathbf{x}} - b_i) = 0$ for $i = 1..m$
- $\widehat{\lambda}_i \geq 0$ for $i = 1..m$
- $A\widehat{\mathbf{x}} \geq \mathbf{b}$... If some $\mathbf{a}_i^T\mathbf{x}^* = b_i$ for some $i \in I^*$ (index set of active constraints) then , one needs to iteratively solve $\mathbf{x}^k$ and I_k
    3. $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{d}^k$
    4. Simplified objective: Find $\mathbf{d}^k = \underset{\mathbf{d}}{\text{argmin}}\ f_k(\mathbf{d})$

Quadratic Optimization: Primal Active-Set Algorithm

$$\mathbf{d}^k = \begin{array}{ll} \text{argmin} & f_k(\mathbf{d}) = \frac{1}{2}\mathbf{d}^T Q\mathbf{d} + \mathbf{g}_k^T\mathbf{d} + c_k \\ \text{subject to} & \mathbf{a}_i\mathbf{d} = 0 \text{ for all } i \in I_k \end{array} \tag{90}$$

where $\mathbf{g}_k = Q\mathbf{x}^k + \mathbf{c}$ and $c_k = (\mathbf{x}^k)^T Q\mathbf{x}^k + \mathbf{c}^T\mathbf{x}^k$. The idea behind the active set algo is:

1. $\mathbf{d}^k = 0 \Rightarrow \mathbf{x}^k$ satisfies first order necessary conditions:
   - $\mathbf{g}^k - \sum_{i\in I_k} \lambda_i \mathbf{a}_i = 0$ which is the same as $rank[A_{\mathcal{I}^k}^T \ \ \mathbf{g}^k] = rank[A_{\mathcal{I}^k}^T]$

   We already know that $\mathbf{a}_i^T\mathbf{x}^k - b_i > 0\, \forall i \notin I_k$ and $\mathbf{a}_i^T\mathbf{x}^k - b_i = 0\, \forall i \in I_k$. Set $\lambda_i = 0\, \forall i \notin I_k$
   1. If $\lambda_i \geq 0\, \forall i \in I_k$, by KKT sufficient conditions, $\mathbf{x}^k$ will be point of global minimum.
   2. If $\lambda_i < 0$ for some $i \in I_k$, then it can be shown that if i is dropped from I_k, the active set and (90) is solved then $\mathbf{d}^k$ will be a descent direction $\nabla^T f(\mathbf{x}^k)\mathbf{d}^k < 0$ and reduce objective
2. $\mathbf{d}^k \neq 0 \ \Rightarrow$ we need to further determine α_k such that $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k\mathbf{d}^k$ remains

   feasible: $\alpha_k = \min \left\{ 1, \ \min_{\substack{j\notin\mathcal{I}^k \\ \mathbf{a}_j^T\mathbf{d}^k<0}} \frac{\mathbf{a}_j^T\mathbf{x}^k - b_j}{-\mathbf{a}_j^T\mathbf{d}^k} \right\}$

Quadratic Optimization: Primal Active-Set Algorithm

Step 1

Input a feasible point, $\mathbf{x}^0$, identify the active set $\mathcal{I}^0$, form matrix $A_{\mathcal{I}^0}$, and set $k = 0$.

Step 2

Compute $\mathbf{g}^k = Q\mathbf{x}^k + \mathbf{c}$.

Check the rank condition $rank[A_{\mathcal{I}^k}^T \;\; \mathbf{g}^k] = rank[A_{\mathcal{I}^k}^T]$. If it does not hold, go to **Step 4**.

Step 3

Solve the system $A_{\mathcal{I}^k}^T \widehat{\lambda} = \mathbf{g}^k$. If $\widehat{\lambda} \geq \mathbf{0}$, output $\mathbf{x}^k$ as the solution and stop; otherwise, remove the index that is associated with the most negative Lagrange multiplier (some $\widehat{\lambda}_t$) from $\mathcal{I}^k$.

Step 4

Compute the value of $\mathbf{d}^k$:

$$\mathbf{d}^k = \begin{array}{ll} \underset{\mathbf{d}}{\operatorname{argmin}} & \frac{1}{2}\mathbf{d}^T Q\mathbf{d} + (\mathbf{g}^k)^T\mathbf{d} \\ \text{subject to} & \mathbf{a}_i^T\mathbf{d} = 0 \qquad \text{for } i \in \mathcal{I}^k \end{array} \tag{91}$$

Step 5

$$\alpha_k = \min \left\{ 1, \min_{\substack{j \notin \mathcal{I}^k \\ \mathbf{a}_j^T \mathbf{d}^k < 0}} \frac{\mathbf{a}_j^T \mathbf{x}^k - b_j}{-\mathbf{a}_j^T \mathbf{d}^k} \right\} \tag{92}$$

Set $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{d}^k$.

Step 6

If $\alpha_k < 1$, construct $\mathcal{I}^{k+1}$ by adding the index that yields the minimum value of α_k in (92). Otherwise, let $\mathcal{I}^{k+1} = \mathcal{I}^k$.

Step 7

Set $k = k + 1$ and repeat from **Step 2**.

Figure 30: Optimization for the quadratic problem in (89) using Primal Active-set Method.

OPTIONAL: Empirical Risk Minimization

Contents

- Learning as mathematical optimization
  - Stochastic optimization, ERM, online regret minimization
  - Offline/online/stochastic gradient descent
- Regularization
  - AdaGrad and optimal regularization
- Gradient Descent++
  - Frank-Wolfe, acceleration, variance reduction, second order methods, non-convex optimization

Recap: Machine Learning as Optimization

$$\widehat{\mathbf{w}}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \; \mathcal{L}(\mathbf{w}) + \Omega(\mathbf{w}) \tag{93}$$

where $\Omega(\mathbf{w})$ is the regularization term.

- **0-1 Loss:**

$$\mathcal{L}(\mathbf{w}) = \sum_{(\mathbf{x},y)} \delta\left(y \neq \mathbf{w}^T \phi(\mathbf{x})\right) \tag{94}$$

  Minimizing the 0-1 Loss is NP-hard. We therefore look for surrogates.

- **Perceptron:** A Non-convex Surrogate

$$\mathcal{L}(\mathbf{w}) = -\sum_{(\mathbf{x},y)\in\mathcal{M}} y\mathbf{w}^T \phi(\mathbf{x}) \tag{95}$$

  where $\mathcal{M} \subseteq \mathcal{D}$ is the set of misclassified examples.

Recap: Convex Surrogates for 0-1 Loss in ML

$$\widehat{\mathbf{w}}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\left(\mathbf{x}^{(i)}, y^{(i)}, \mathbf{w}\right) + \Omega(\mathbf{w}) \tag{96}$$

- **Logistic Regression:**

$$\mathcal{L}\left(\mathbf{x}^{(i)}, y^{(i)}, \mathbf{w}\right) = -\left[\left(y^{(i)} \mathbf{w}^T \phi(\mathbf{x}^{(i)}) - \log\left(1 + exp\left(\mathbf{w}^T \phi\left(\mathbf{x}^{(i)}\right)\right)\right)\right)\right] \tag{97}$$

- **Sigmoidal Neural Net:**

$$\mathcal{L}(\mathbf{w}) = -\frac{1}{m}\left[\sum_{i=1}^{m} \sum_{k=1}^{K} y_k^{(i)} \log\left(\sigma_k^L\left(\mathbf{x}^{(i)}\right)\right) + \left(1 - y_k^{(i)}\right) \log\left(1 - \sigma_k^L\left(\mathbf{x}^{(i)}\right)\right)\right] \tag{98}$$

Recap: Convex Surrogates for 0-1 Loss in ML

$$\widehat{\mathbf{w}}^* = \underset{\mathbf{w}}{\text{argmin}}\ \mathcal{L}(\mathbf{w}) + \Omega(\mathbf{w}) \tag{99}$$

- **Logistic Regression:**

$$\mathcal{L}(\mathbf{w}) = -\left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\mathbf{w}^T\phi(\mathbf{x}^{(i)}) - \log\left(1 + exp\left(\mathbf{w}^T\phi\left(\mathbf{x}^{(i)}\right)\right)\right)\right)\right] \tag{100}$$

- **Sigmoidal Neural Net:**

$$\mathcal{L}(\mathbf{w}) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(\sigma_k^L\left(\mathbf{x}^{(i)}\right)\right) + \left(1 - y_k^{(i)}\right)\log\left(1 - \sigma_k^L\left(\mathbf{x}^{(i)}\right)\right)\right] \tag{101}$$

Empirical Risk Minimization and Projected Gradient Descent

Empirical Risk Minimization and Proj Grad Descent

- Gradient depends on all data
- What about generalization?
- Simultaneous optimization and generalization
  - Faster optimization! (single example per iteration)

Statistical (PAC) learning

- $\mathcal{D}$: i.i.d distribution over $\mathcal{X} \times \mathcal{Y} = \{(\mathbf{x}^i, y^i)\}$
- Goal: To learn Hypothesis h from hypothesis class $\mathcal{H}$ that minimizes expected loss $err(h) = \mathbf{E}\left[\mathcal{L}(\mathbf{x}^i, y^i, \mathbf{w})\right]$.
- $\mathcal{H}$ is (PAC) learnable if $\forall \epsilon, \delta > 0$, there exists algorithm s.t. after seeing M examples, where $M = \mathcal{O}\left(poly(\delta, \epsilon, dimension(\mathcal{H}))\right)$, the algorithm finds h s.t. w.p. $1 - \delta$,

$$err(h) \leq \min_{h^* \in \mathcal{H}} err(h^*) + \epsilon$$

Online Learning and Regret Minimization

- For $k = 1, 2 \ldots K$, $h^k \in \mathcal{H}$, and an adversarial example $(\mathbf{x}^k, y^k)$, minimize expected regret:

$$\frac{1}{K}\left[\sum_k \mathcal{L}(h^k, \mathbf{x}^k, y^k) - \min_{h^* \in \mathcal{H}} \sum_k \mathcal{L}(h^*, \mathbf{x}^k, y^k)\right] \xrightarrow{K \to \infty} 0$$

- Generalization in PAC setting is achieved by regret vanishing

Online Gradient Descent: Efficient Algorithm for Regret Minimization

- Let us denote by ∇_k, the expression $\nabla_{\mathbf{w}^k}\mathcal{L}\left(\mathbf{x}^k, y^k, \mathbf{w}^k\right)$
- Note that some adversarial example $(\mathbf{x}^k, y^k)$ could be the same as $(\mathbf{x}^l, y^l)$ for $l \neq k$
- The alternating steps are
  - Stochastic gradient descent Step: $\mathbf{w}_u^{k+1} = \mathbf{w}_p^k - t\nabla_k$
  - Projection Step: $\mathbf{w}_p^{k+1} = \underset{z \in \mathcal{C}}{\operatorname{argmin}} \|\mathbf{w}_u^k - z\|$
- **Claim:** Regret $= \sum_{k=1}^{K} \mathcal{L}(\mathbf{x}^k, y^k, \mathbf{w}^k) - \sum_{k=1}^{K} \mathcal{L}(\mathbf{x}^k, y^k, \mathbf{w}^*) = \mathcal{O}(K)$

Online Gradient Descent: Analysis

- Online Gradient Descent: Efficient Algorithm for Regret Minimization - Zinkevich 2005
- As before, substituting for $\mathbf{w}_u^{k+1}$ and expanding squares

$$\|\mathbf{w}_u^{k+1} - \mathbf{w}^*\|^2 = \|\mathbf{w}_p^k - \mathbf{w}^*\|^2 - 2t\nabla_k(\mathbf{w}^* - \mathbf{w}_p^k) + t^2\|\nabla_k\|^2 \tag{102}$$

- Since $\mathbf{w}_p^{k+1} = \underset{z \in \mathcal{C}}{\text{argmin}} \ \|\mathbf{w}_u^k - z\|$,

$$\|\mathbf{w}_p^{k+1} - \mathbf{w}^*\|^2 \leq \|\mathbf{w}_u^{k+1} - \mathbf{w}^*\|^2 \tag{103}$$

- Substituting from equality (102) into the RHS of inequality (103):

$$\|\mathbf{w}_p^{k+1} - \mathbf{w}^*\|^2 \leq \|\mathbf{w}_p^k - \mathbf{w}^*\|^2 - 2t\nabla_k(\mathbf{w}_p^k - \mathbf{w}^*) + t^2\|\nabla_k\|^2 \tag{104}$$

- By convexity,

$$\sum_{k=1}^{K} \mathcal{L}(\mathbf{x}^k, y^k, \mathbf{w}_p^k) - \mathcal{L}(\mathbf{x}^k, y^k, \mathbf{w}^*) \leq \sum_{k=1}^{K} \nabla_k(\mathbf{w}^* - \mathbf{w}_p^k) \tag{105}$$

Online Gradient Descent: Analysis (contd)

- Substituting from (104) into (105)

$$\sum_{k=1}^{K} \mathcal{L}(\mathbf{x}^k, y^k, \mathbf{w}_p^k) - \mathcal{L}(\mathbf{x}^k, y^k, \mathbf{w}^*) \leq \sum_{k=1}^{K} \frac{1}{2t} \left(\|\mathbf{w}_p^k - \mathbf{w}^*\|^2 - \|\mathbf{w}_p^{k+1} - \mathbf{w}^*\|^2 + t^2 \|\nabla_k\|^2 \right) \tag{106}$$

- As before, if: $\mathbf{g}$ is upper bound on norm of gradients, *i.e.*, $\|\nabla f(x)\|^2 \leq \mathbf{g}^2$
- Using the above upper bound and expanding the summation over $\|\mathbf{w}^* - \mathbf{w}^k\|^2$, all terms get canceled except for the first and last:

$$\sum_{k=1}^{K} \mathcal{L}(\mathbf{x}^k, y^k, \mathbf{w}_p^k) - \mathcal{L}(\mathbf{x}^k, y^k, \mathbf{w}^*) \leq \frac{1}{2t} \left(\|\mathbf{w}_p^1 - \mathbf{w}^*\|^2 - \|\mathbf{w}_p^{K+1} - \mathbf{w}^*\|^2 \right) + \frac{t}{2} K \mathbf{g}^2 \tag{107}$$

- Using the fact that negative of norm is always negative

$$\sum_{k=1}^{K} \mathcal{L}(\mathbf{x}^k, y^k, \mathbf{w}_p^k) - \mathcal{L}(\mathbf{x}^k, y^k, \mathbf{w}^*) \leq \frac{1}{2t} \left(\|\mathbf{w}_p^1 - \mathbf{w}^*\|^2 \right) + \frac{t}{2} K \mathbf{g}^2 \tag{108}$$

Online Gradient Descent: Analysis (contd)

- Again recall that $\mathbf{d}$ is diameter of $\mathcal{C}$, *i.e.*, $\mathbf{w} \in \mathcal{C}$, $\|\mathbf{w}_p^1 - \mathbf{w}^*\|^2 \leq \mathbf{d}^2$, thus, (108) becomes (109)

$$\sum_{k=1}^{K} \mathcal{L}(\mathbf{x}^k, y^k, \mathbf{w}_p^k) - \mathcal{L}(\mathbf{x}^k, y^k, \mathbf{w}^*) \leq \frac{\mathbf{d}^2}{2t} + \frac{t}{2} K \mathbf{g}^2 \tag{109}$$

- Since $\frac{\mathbf{d}^2}{2t} + \frac{t}{2} K\mathbf{g}^2 = \frac{\mathbf{d}^2}{2t} + \frac{t}{2} K\mathbf{g}^2 - \mathbf{g}\mathbf{d}\sqrt{K} + \mathbf{g}\mathbf{d}\sqrt{K} = \left(\frac{\mathbf{d}}{\sqrt{2t}} - \sqrt{\frac{Kt}{2}}\mathbf{g}\right)^2 + \mathbf{g}\mathbf{d}\sqrt{K} \geq \mathbf{g}\mathbf{d}\sqrt{K}$ and therefore,

$$\sum_{k=1}^{K} \mathcal{L}(\mathbf{x}^k, y^k, \mathbf{w}_p^k) - \mathcal{L}(\mathbf{x}^k, y^k, \mathbf{w}^*) \leq \mathbf{g}\mathbf{d}\sqrt{K} = \Omega(\sqrt{K}) \tag{110}$$

- Thus, Regret $= \Omega(\sqrt{K})$

- Based on the derivations starting from (105) that culminate in (110), we now know that

$$\sum_{k=1}^{K} \nabla_k(\mathbf{w}_p^k - \mathbf{w}^*) \leq \mathbf{gd}\sqrt{K} \tag{111}$$

- Thus,

$$\frac{1}{K}\sum_{k=1}^{K} \nabla_k(\mathbf{w}_p^k) = \frac{1}{K}\sum_{k=1}^{K} \nabla_k(\mathbf{w}_p^k) + \frac{\mathbf{gd}}{\sqrt{K}} \tag{112}$$

- Treating each $(\mathbf{x}^k, y^k)$ to be a random example and taking expectations over such samples $(\mathbf{x}^k, y^k)$ while combining (111) and (106)

$$\mathbf{E}\left[\frac{1}{K}\sum_{k=1}^{K} \mathcal{L}(\mathbf{x}^k, y^k, \mathbf{w}_p^k) - \mathcal{L}(\mathbf{x}^k, y^k, \mathbf{w}^*)\right] \leq \mathbf{E}\left[\frac{1}{K}\sum_{k=1}^{K} \nabla_k(\mathbf{w}_p^k - \mathbf{w}^*)\right] \leq \mathbf{E}\left[\frac{\mathbf{gd}}{\sqrt{K}}\right] \tag{113}$$

Summarizing Analysis for Stochastic Gradient Descent

- One example per step, same convergence properties as projected gradient descent and additional provides **direct generalization**! (All this formally needs martingales)

$$\mathbf{E}\left[\frac{1}{K}\sum_{k=1}^{K}\mathcal{L}(\mathbf{x}^k, y^k, \mathbf{w}_p^k) - \mathcal{L}(\mathbf{x}^k, y^k, \mathbf{w}^*)\right] \leq \mathbf{E}\left[\frac{1}{K}\sum_{k=1}^{K}\nabla_k(\mathbf{w}_p^k - \mathbf{w}^*)\right] \leq \mathbf{E}\left[\frac{\mathbf{gd}}{\sqrt{K}}\right]$$

- To get solution that is ϵ approximate with $\epsilon = \frac{\mathbf{dg}}{\sqrt{K}}$, you need number of gradient iterations that is $K = \left(\frac{\mathbf{dg}}{\epsilon}\right)^2 = O\left(\frac{1}{\epsilon}\right)^2$
- Recall that $\mathcal{H}$ is (PAC) learnable if $\forall\epsilon, \delta > 0$, there exists algorithm s.t. after seeing M examples, where $M = \mathcal{O}\left(poly(\delta, \epsilon, dimension(\mathcal{H}))\right)$, the algorithm finds h s.t. w.p. $1-\delta$,

$$err(h) \leq \min_{h^* \in \mathcal{H}} err(h^*) + \epsilon$$

- Thus, the number of iterations for ϵ approximation is $K = M\left(\frac{\mathbf{dg}}{\epsilon}\right)^2 = O\left(\frac{M}{\epsilon}\right)^2$

Follow the Leader

- Recap (slightly different) definition of regret:

$$\sum_{k=1}^{K} \mathcal{L}(\mathbf{x}^k, y^k, \mathbf{w}_p^k) - \min_{\mathbf{w} \in \mathcal{C}} \sum_{k=1}^{K} \mathcal{L}(\mathbf{x}^k, y^k, \mathbf{w}) \tag{114}$$

- Minimizing regret might still not show stability wrt $|\mathbf{w}^{k+1} - \mathbf{w}^k|$. Eg: When $+1$ and -1 are alternating!
- Consider Follow-The-Leader (FTL or best-in-hindsight) that minimizes a linear approximation of the loss function:

$$\mathbf{w}^k = \operatorname*{argmin}_{\mathbf{w} \in \mathcal{C}} \sum_{i=1}^{k-1} \mathbf{w}^T \nabla \mathcal{L}(\mathbf{x}^i, y^i, \mathbf{w}^i)$$

Regularizing Follow the Leader

- Given Follow-The-Leader (FTL)....

$$\mathbf{w}^k = \operatorname*{argmin}_{\mathbf{w} \in \mathcal{C}} \sum_{i=1}^{k-1} \mathbf{w}^T \nabla \mathcal{L}(\mathbf{x}^i, y^i, \mathbf{w}^i)$$

- ....Follow-The-Regularized-Leader (FTRL) additionally regularizes this loss function

$$\mathbf{w}^k = \operatorname*{argmin}_{\mathbf{w} \in \mathcal{C}} \sum_{i=1}^{k-1} \mathbf{w}^T \nabla \mathcal{L}(\mathbf{x}^i, y^i, \mathbf{w}^i) + \frac{1}{t}\Omega(\mathbf{w})$$

- $\Omega(\mathbf{w})$ is often chosen to be a strongly convex function in order to ensure stability (Kalai Vempala observation):

$$\nabla \mathcal{L}(\mathbf{x}^i, y^i, \mathbf{w}^k) = \mathcal{O}(t)$$

- Perspectives for regularization
  1. PAC theory: Reduce complexity
  2. Regret Minimization: Improve Stability

FTRL *i.e.*, Mirror Descent

- Follow-The-Regularized-Leader (FTRL):

$$\mathbf{w}^k = \underset{\mathbf{w} \in \mathcal{C}}{\text{argmin}} \sum_{i=1}^{k-1} \mathbf{w}^T \nabla \mathcal{L}(\mathbf{x}^i, y^i, \mathbf{w}^i) + \frac{1}{t}\Omega(\mathbf{w})$$

- Bregman Divergence, another perspective that gives you generalized regret bounds:

$$B_\Omega(\mathbf{w}_p || \mathbf{w}_u) = \Omega(\mathbf{w}_p) - \Omega(\mathbf{w}_u) - (\mathbf{w}_p - \mathbf{w}_u)^t \nabla \Omega(\mathbf{w}_u)$$

- Consider the Bregman Projection:

$$P_{\mathcal{C}}^{\Omega}(\mathbf{w}_u) = \underset{\mathbf{w}_p \in \mathcal{C}}{\arg\min} \; B_\Omega(\mathbf{w}_p || \mathbf{w}_u)$$

- The Online Mirror Descent Algorithm with following steps is equivalent to FTRL:
  1. $\mathbf{w}^k \equiv \mathbf{w}_p^k = P_{\mathcal{C}}^{\Omega}(\mathbf{w}_u^k)$
  2. $\mathbf{w}_u^{k+1} = (\nabla \Omega)^{-1}(\nabla \Omega(\mathbf{w}_u^k) - t \nabla \mathcal{L}(\mathbf{x}^i, y^i, \mathbf{w}_p^k)$

Eg: $\Omega(\mathbf{w}) = \|\mathbf{w}\|^2$

- Follow-The-Regularized-Leader (FTRL):

$$\mathbf{w}^k = P_{\mathcal{C}}\left(-t\sum_{i=1}^{k-1}\nabla\mathcal{L}(\mathbf{x}^i, y^i, \mathbf{w})\right)$$

- Bregman Divergence:

$$B_\Omega(\mathbf{w}_p||\mathbf{w}_u) = \|\mathbf{w}_p\|^2 - \|\mathbf{w}_u\|^2 - 2(\mathbf{w}_p - \mathbf{w}_u)^t\mathbf{w}_u = \|\mathbf{w}_p - \mathbf{w}_u\|^2$$

- The Online Mirror Descent Algorithm:
  1. $\mathbf{w}_p^k = \text{argmin}_{\mathbf{w}_p\in\mathcal{C}}\ \|\mathbf{w}_p - \mathbf{w}_u^k\|^2$
  2. $\mathbf{w}_u^{k+1} = (\nabla\Omega)^{-1}\left(2\mathbf{w}_u^k - t\nabla\mathcal{L}(\mathbf{x}^i, y^i, \mathbf{w}_p^k)\right)$
- Thus turns out to be ordinary projected gradient descent!

Eg: $\Omega(\mathbf{w}) = \sum_j w_j \log w_j$

- Additionally require a loss linear in $\mathbf{w}$: $\mathcal{L}(\mathbf{x}^i, y^i, \mathbf{w}) = \mathbf{w}^T\mathbf{c}^i$ where $\mathbf{c}^i$ is a vector of losses.
- Follow-The-Regularized-Leader (FTRL) with the normalization factor Z_k being a function of $\mathcal{C}$:

$$\mathbf{w}^k = \frac{\exp\left(-t\sum_{i=1}^{k-1}\right)}{Z_k}$$

- Bregman Divergence:

$$B_\Omega(\mathbf{w}_p||\mathbf{w}_u) = \sum_j \left[(\mathbf{w}_p)_j \log(\mathbf{w}_p)_j - (\mathbf{w}_u)_j \log(\mathbf{w}_u)_j - ((\mathbf{w}_p)_j - (\mathbf{w}_u)_j)(\log(\mathbf{w}_u)_j + 1)\right] \quad (115)$$

$$= \sum_j \left[(\mathbf{w}_p)_j \log(\mathbf{w}_p)_j - (\mathbf{w}_p)_j \log(\mathbf{w}_u)_j - ((\mathbf{w}_p)_j - (\mathbf{w}_u)_j)\right] \quad (116)$$

- The Online Mirror Descent Algorithm:
  1. $\mathbf{w}_p^k = \text{argmin}_{\mathbf{w}_p \in \mathcal{C}} \ \sum_j \left[(\mathbf{w}_p^k)_j \log \frac{(\mathbf{w}_p^k)_j}{e \times (\mathbf{w}_u^k)_j}\right]$
  2. $\mathbf{w}_u^k + 1 = (\nabla\Omega)^{-1}\left(\log \mathbf{w}_u^k - t\nabla\mathcal{L}(\mathbf{x}^i, y^i, \mathbf{w}_p^k)\right)$

Adaptive Regularization: Adagrad

- The general regularized follow the leader (RFTL):

$$\mathbf{w}^k = \operatorname*{argmin}_{\mathbf{w} \in \mathcal{C}} \sum_{i=1}^{k-1} \mathcal{L}(\mathbf{x}^i, y^i, \mathbf{w}^i) + \frac{1}{t}\Omega(\mathbf{w})$$

- A natural question is, which $\Omega(\mathbf{w})$ to pick? Solution: Learn!!
- Adagrad: Learn to pick from a family of regularizers

$$\Omega(\mathbf{w}) = |\mathbf{w}|_R^2 \ \mathbf{s}.t.\ R \geq 0,\ \ Trace(R) = \omega$$

Adaptive Regularization: Adagrad (contd.)

- Set $\mathbf{w}^1$ arbitrarily
- For $k = 1, 2, \ldots$
  1. Compute $\mathcal{L}(\mathbf{x}^k, y^k, \mathbf{w}^k)$
  2. Compute $\mathbf{w}^{(k+1)} = \mathbf{w}_p^{(k+1)}$ as follows:
     - ★ $H_k = diag(\sum_{i=1}^k \nabla\mathcal{L}(\mathbf{x}^k, y^k, \mathbf{w}^k)\mathcal{L}(\mathbf{x}^k, y^k, \mathbf{w}^k)^T)$
     - ★ $\mathbf{w}_u^{(k+1)} = \mathbf{w}^k - tH_k^{\frac{-1}{2}} \nabla\mathcal{L}(\mathbf{x}^k, y^k, \mathbf{w}^k)$
     - ★ $\mathbf{w}_p^{(k+1)} = \operatorname*{argmin}_{\mathbf{w}\in\mathcal{C}} (\mathbf{w}_u^{(k+1)} - \mathbf{w})^T H_k(\mathbf{x}_u^{k+1} - \mathbf{w})$
- Regret Bound: $\mathcal{O}\left(\sum_i \sqrt{\sum_k \nabla\mathcal{L}(\mathbf{x}^i, y^i, \mathbf{w}^k)}\right)$ can be $\sqrt{d}$ better than Stochastic Gradient Descent
- Infrequently occurring, or small-scale, features have small influence on regret (and therefore, convergence to optimal parameter)

Accelerating Gradient Descent: Variance Reduction

- Uses the special structure of Empirical Risk Minimization
- Very effective for Lipschitz continuous (smooth) & convex functions
- Recap: Condition number of Convex Functions $= \frac{L}{\alpha} =$ Ratio of Lipschitz constant (L) and strong convexity factor (α)

$$0 \prec \alpha I \preceq \nabla^2 f(\mathbf{x}) \preceq LI$$