Coordinate Optimization

i) Choose coordinate j

ii) Update value at coordinate j

$$x^{t+1} = x^{t} - \alpha_t \nabla f(x^t) \cdot e_{j_t}$$

faster if updates are d times faster

$$\nabla f(x^t) \cdot e_{j_t} \in O(nd)$$

2.1.1

compute $\nabla_j f(x^t) \in O(n)$

- need $X \cdot w$ to compute $\nabla_j f(x^t)$

  $X \cdot w \in O(n \cdot d)$

- can update Xw based on how w changes at each iteration $\in O(n)$

2.1.2

$L_c = \max_j \{L_j\}$ since L is an upper bound

SampleDiscrete(p)

$\sum_i p_i = 1$ e.g. $p = [0.2 \ 0.3 \ 0.5]$ returns $\begin{cases} 1 \text{ with prob. } 20\% \\ 2 \text{ " " } 30\% \\ 3 \text{ " " } 50\% \end{cases}$

Proximal Gradient

For solving problems of the form:

$$\underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \underbrace{f(x)}_{\text{smooth}} + \underbrace{r(x)}_{\text{non-smooth}}$$

we use updates

$$x^{t+1} = \underset{y \in \mathbb{R}^d}{\operatorname{argmin}} \left\{ \frac{1}{2} \| y - (x^t - \alpha_t \nabla f(x^t)) \|^2 + \alpha_t r(y) \right\}$$

$$= \operatorname{prox}_{\alpha_t r} \left[x^t - \alpha_t \nabla f(x^t) \right]$$

2.2.1

$$\underset{X}{\underbrace{\begin{bmatrix} — x^1 — \\ \vdots \\ — x^n — \end{bmatrix}}_{n \times d}} \underset{W}{\underbrace{\begin{bmatrix} | & & | \\ w_1 & \cdots & w_k \\ | & & | \end{bmatrix}}_{d \times k}} = \underset{Y}{\underbrace{\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}}_{n \times k}}$$

$$f(w) = \sum_i -\log \frac{e^{w_{y_i}^T x_i}}{\sum_k e^{w_k^T x_i}} + \underline{\qquad\qquad}$$

Add L2-regularization to all and gradient (Use Frobenius norm)

$$g(w) = \ldots + \underline{\qquad\qquad}$$

2.2.2

Instead of findMin, use proxGradL1(f, w, λ, ...)

↳ returns $\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} f(w) + \lambda \|w\|_1$

- Just pass in the differentiable part of the function (and gradient for that part) - ~~thoseofunctions~~ proxGradL1 does the regularization by applying the proximal operator for L1-regularization
  - which is an element-wise soft threshold

$$x_t = \frac{x_t}{|x_t|} \max\{0, |x_t| - \alpha_t \lambda\}$$

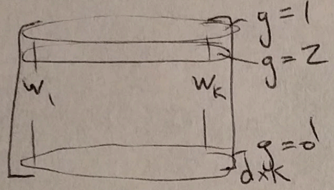
2.2.3

Modify the proximal operator to be the proximal operator for <u>group</u> L1 regularization

- which is a group-wise soft threshold

$$x_g = \frac{x_g}{\|x_g\|_2} \max\{0, \|x_g\|_2 - \alpha_t \lambda\}$$

- Add a parameter to identify which group each x_t is in.
- Declare the value in softmaxClassifierGL1 and pass it to proxGradGL1



0 rows in W correspond to unused features (columns of X)

x_g: row of W

Stochastic Gradient

gradient wrt training example i

$$x^{t+1} = x^t - \frac{\alpha_t}{n} \sum_i \nabla^i f(x^t)$$

Stochastic: $x^{t+1} = x^t - \alpha_t \nabla^{i_t} f(x^t)$

gradient wrt random training example i_t

$\Rightarrow O(n)$ faster per iteration

2.3.3

AdaGrad - want different step sizes for different dimensions

$$x^{t+1} = x^t - \alpha_t D_t \nabla f(x^t)$$

Diagonal Matrix

$$D(i,i) = \frac{1}{\sqrt{\delta + \sum_{k=0}^{t} (\nabla_i f(x^k))^2}}$$

2.3.4

S.A.G. - keep memory of gradients w.r.t each training example
- update 1 gradient per iteration

$$G = \begin{bmatrix} g^1 \\ g^2 \\ \vdots \\ g^n \end{bmatrix}$$

At each iteration

$i \leftarrow$ random training index

$g^i \leftarrow \nabla^i f(x^t)$

$x^{t+1} \leftarrow x^t - \frac{\alpha_t}{n} \sum_i g^i$

$\hookrightarrow$ Can initialize G with all zeros and still use n at every iteration