Coordinate Optimization

i) Choose coordinate $j$

ii) Update value at coordinate $j$

$$x^{t+1} = x^t - \lambda_t \nabla f(x^t) \cdot e_j$$

Faster updates are $d$ times faster.

$$\nabla f(x^t) \cdot e_j \in O(nd)$$

2.1.1

Compute $\nabla f(x^t) \in O(n)$

- Need $Xw$ to compute $\nabla f(x^t)$

$$Xw \in O(n \cdot d)$$

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

2.1.2

$$L_c = \max \{ \frac{3}{4}, 1 \}$$

since $L$ is an upper bound.

Sample Discrete ($p$)

$$\sum_i p_i = 1 \quad \text{e.g. } \ p = [0.2, 0.3, 0.5]$$

returns

1 with prob. 20%  
2 with 30%  
3 with 50%
Proximal Gradient

For solving problems of the form:

\[
\text{argmin}_{x \in \mathbb{R}^d} \; f(x) + r(x)
\]

where we use updates

\[
x_{t+1} = \text{prox}_\gamma \left( x_t - \frac{\lambda}{\gamma} \nabla f(x_t) \right)
\]

2.2.1

\[
\begin{bmatrix}
  x^1 \\
  \vdots \\
  x^n
\end{bmatrix}
\begin{bmatrix}
  w_1 \\
  \vdots \\
  w_k
\end{bmatrix}
= \begin{bmatrix}
  y
\end{bmatrix}
\]

\[
\begin{align*}
  f(w) &= \sum_i -\log \frac{e^{w_i x_i}}{\sum_k e^{w_k x_i}} \\
  g(w) &= \text{...}
\end{align*}
\]

2.2.2

Instead of \text{gradMin}, use \text{proxGradL1}(f, w, \lambda, \ldots)

\[
\text{L1-returns argmin}_{x \in \mathbb{R}^d} \; f(x) + \|x\|_1
\]

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

\[
x_t^+ = \frac{x^+}{|x^+|} \max \{ 0, 1 - |x_t| \}
\]
2.2.3

Modify the proximal operator to be the proximal operator for group L1 regularization, which is a group-wise soft threshold:

\[ x_q = \frac{x_q}{\|x_q\|_2} \max \left\{ 0, \|x_q\|_2 - \lambda_e \right\} \]

- Add a parameter to identify which group each \( x_q \) is in.
- Declare the value in softmax classifier G1L and pass it to proxGradGL1

\[ \text{0 rows in } W \text{ correspond to unused features (columns of } x) \]

**Stochastic Gradient**

\[ x^{(t+1)} = x^t - \frac{\lambda_e}{n} \sum_i \nabla_i f(x^t) \]

**Stochastic**: 

\[ x^{(t+1)} = x^t - \frac{\lambda_e}{n} \nabla f(x^t) \]

(gradient w.r.t training example \( i \))

\[ \Rightarrow O(n) \text{ faster per iteration} \]

2.3.3

AdaGrad - want different step sizes for different dimensions

\[ x^{(t+1)} = x^t - \lambda_e \nabla f(x^t) \]

\( O(\mathbf{z}, i) = \frac{1}{\sqrt{\sum \left( \nabla f(x^t) \right)^2}} \)
2.3.4

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

At each iteration

1. \( i \leftarrow \text{random training index} \)
2. \( g^i = \nabla f(x^i) \)
3. \( x^{t+1} = x^t - \frac{x^t}{n} \sum_{j} g^j \)

\( \Rightarrow \) Can initialize \( G \) with all zeros and still use \( n \) at every iteration