LEARNING AS OPTIMIZATION: MOTIVATION

Learning as optimization: general procedure

- Goal: Learn parameter **θ** (or weight vector **w**)
- Dataset: $D=\{(x_1,y_1)\ldots,(x_n, y_n)\}$
- Write down loss function: how well **w** fits the data D as a function of **w**
  - Common choice: log Pr(D|**w**)
- Maximize by differentiating
  - Then **gradient descent**: repeatedly take a small step in the direction of the gradient

Learning as optimization: general procedure for SGD (stochastic gradient descent)

- **Big-data** problem: we *don't* want to load all the data *D* into memory, and the gradient depends on all the data
- **Solution:**
  - pick a small subset of examples B<<D
  - **approximate** the gradient using them
    - "on average" this is the right direction
  - take a step in that direction
  - repeat….
- Math: find gradient of **w** for a *single* example, not a dataset

B = **one** example is a very popular choice

SGD vs streaming

- Streaming:
  - pass through the data *once*
  - hold model + one example in memory
  - update model for each example
- Stochastic gradient:
  - pass through the data *multiple times*
    - stream through a disk file repeatedly
  - hold model + B examples in memory
  - update model *via gradient step*

B = **one** example is a very popular choice

its simple ☺

sometimes its cheaper to evaluate 100 examples at once than one example 100 times ☹

Efficient Logistic Regression with Stochastic Gradient Descent

William Cohen

Learning as optimization for logistic regression

- Goal: Learn the parameter **w** of the classifier

$$P(Y=y|X=\mathbf{x},\mathbf{w}) = \frac{1}{1+e^{-\mathbf{x}\cdot\mathbf{w}}}$$

- Probability of a single example P(y|**x,**w) would be

$$P(Y=y|X=\mathbf{x},\mathbf{w}) = \begin{cases} \frac{1}{1+e^{-\mathbf{x}\cdot\mathbf{w}}} & \text{if } y=1 \\ 1-\frac{1}{1+e^{-\mathbf{x}\cdot\mathbf{w}}} & \text{if } y=0 \end{cases}$$

- Or with logs:

$$\log P(Y=y|X=\mathbf{x},\mathbf{w}) = \begin{cases} \log p & \text{if } y=1 \\ \log(1-p) & \text{if } y=0 \end{cases}$$

$$p \equiv \frac{1}{1+e^{-\mathbf{x}\cdot\mathbf{w}}} = \frac{1}{1+\exp(-\sum_j x^j w^j)}$$

$$\log P(Y=y|X=\mathbf{x},\mathbf{w}) = \begin{cases} \log p & \text{if } y=1 \\ \log(1-p) & \text{if } y=0 \end{cases}$$

$$\frac{\partial}{\partial w^j}\log P(Y=y|X=\mathbf{x},\mathbf{w}) = \begin{cases} \frac{1}{p}\boxed{\frac{\partial}{\partial w^j}p} & \text{if } y=1 \\ \frac{1}{1-p}\left(-\frac{\partial}{\partial w^j}p\right) & \text{if } y=0 \end{cases}$$

$$(\log f)' = \tfrac{1}{f}f'$$

$$p \equiv \frac{1}{1+e^{-\mathbf{x}\cdot\mathbf{w}}} = \frac{1}{1+\exp(-\sum_j x^j w^j)}$$

$$p \equiv \frac{1}{1+e^{-\mathbf{x}\cdot\mathbf{w}}} = \frac{1}{1+\exp(-\sum_j x^j w^j)}$$

$$1 - p = \frac{1+\exp(-\sum_j x^j w^j)}{1+\exp(-\sum_j x^j w^j)} - \frac{1}{1+\exp(-\sum_j x^j w^j)} = \boxed{\frac{\exp(-\sum_j x^j w^j)}{1+\exp(-\sum_j x^j w^j)}}$$

$$\boxed{\frac{\partial}{\partial w^j} p} = \frac{\partial}{\partial w^j}(1+\exp(-\sum_j x^j w^j))^{-1} \qquad (e^f)' = e^f f'$$

$$= (-1)(1+\exp(-\sum_j x^j w^j))^{-2} \frac{\partial}{\partial w^j} \exp(-\sum_j x^j w^j)$$

$$= (-1)(1+\exp(-\sum_j x^j w^j))^{-2} \exp(-\sum_j x^j w^j)(-x^j)$$

p

$$= \boxed{\frac{1}{1+\exp(-\sum_j x^j w^j)}} \boxed{\frac{\exp(-\sum_j x^j w^j)}{1+\exp(-\sum_j x^j w^j)}} x^j$$

$$\frac{\partial}{\partial w^j} p = p(1-p)x^j$$

$$\log P(Y=y|X=\mathbf{x},\mathbf{w}) = \begin{cases} \log p & \text{if } y=1 \\ \log(1-p) & \text{if } y=0 \end{cases}$$

$$\frac{\partial}{\partial w^j}\log P(Y=y|X=\mathbf{x},\mathbf{w}) = \begin{cases} \frac{1}{p}\boxed{\frac{\partial}{\partial w^j}p} & \text{if } y=1 \\ \frac{1}{1-p}(-\frac{\partial}{\partial w^j}p) & \text{if } y=0 \end{cases}$$

$$\frac{\partial}{\partial w^j}p = p(1-p)x^j$$

$$\frac{\partial}{\partial w^j}\log P(Y=y|X=\mathbf{x},\mathbf{w}) = \begin{cases} \frac{1}{p}p(1-p)x^j = (1-p)x^j & \text{if } y=1 \\ \frac{1}{1-p}(-1)p(1-p)x^j = -px^j & \text{if } y=0 \end{cases}$$

$$\frac{\partial}{\partial w^j}\log P(Y=y|X=\mathbf{x},\mathbf{w}) = (y-p)x^j$$

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \lambda(y-p)\mathbf{x}$$

$$\log P(Y = y | X = \mathbf{x}, \mathbf{w}) = \begin{cases} \log p & \text{if } y = 1 \\ \log(1-p) & \text{if } y = 0 \end{cases}$$

$$p = \sigma(\mathbf{x} \cdot \mathbf{w})$$

Magically, when we differentiate, we end up with something very simple and elegant.....

$$\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w} \mid y, \mathbf{x}) = (y - p)\mathbf{x}$$

$$\frac{\partial}{\partial w^j} L(\mathbf{w} \mid y, \mathbf{x}) = (y - p)x^j$$

The update for gradient descent is just:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \lambda(y - p)\mathbf{x}$$

Logistic regression has a sparse update

An observation: sparsity!

$$\frac{\partial}{\partial w^j} \log P(Y = y | X = \mathbf{x}, \mathbf{w}) = (y - p) x^j$$

Key computational point:

- if $x^j = 0$ then the gradient of w^j is zero
- so when processing an example you only need to update weights for the **non-zero** features of an example.

Learning as optimization for logistic regression

- The algorithm: $$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \lambda(y - p)\mathbf{x}$$

1. Initialize a hashtable W
2. For $t = 1, \ldots, T$
   - For each example $\mathbf{x}_i, y_i$: *-- do this in random order*
     - Compute the prediction for $\mathbf{x}_i$:
       $$p_i = \frac{1}{1 + \exp(-\sum_{j:x_i^j > 0} x_i^j w^j)}$$
     - For each non-zero feature of $\mathbf{x_i}$ with index j and value x^j:
       * If j is not in W, set $W[j] = 0$.
       * Set $W[j] = W[j] + \lambda(y_i - p_i)x^j$
3. Output the hash table W.

REGULARIZED LOGISTIC REGRESSION

Regularized logistic regression

- Replace LCL

$$\log P(Y=y|X=\mathbf{x},\mathbf{w}) = \begin{cases} \log p & \text{if } y=1 \\ \log(1-p) & \text{if } y=0 \end{cases}$$

- with LCL + penalty for large weights, eg

$$LCL - \mu \sum_{j=1}^{d} (w^j)^2$$

- So:

$$\frac{\partial}{\partial w^j} \log P(Y=y|X=\mathbf{x},\mathbf{w}) = (y-p)x^j$$

- becomes:

$$\frac{\partial}{\partial w^j} \log P(Y=y|X=\mathbf{x},\mathbf{w}) - \mu \sum_{j=1}^{d} (w^j)^2 = (y-p)x^j - 2\mu w^j$$

Regularized logistic regression

- Replace LCL

$$\log P(Y = y|X = \mathbf{x}, \mathbf{w}) = \begin{cases} \log p & \text{if } y = 1 \\ \log(1-p) & \text{if } y = 0 \end{cases}$$

- with LCL + penalty for large weights, eg

$$LCL - \mu \sum_{j=1}^{d} (w^j)^2$$

- So the update for wj becomes:

$$w^j = w^j + \lambda((y-p)x^j - 2\mu w^j)$$

- Or

$$w^j = w^j + \lambda(y-p)x^j - \lambda 2\mu w^j$$

Learning as optimization for logistic regression

- Algorithm:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \lambda(y - p)\mathbf{x}$$

1. Initialize a hashtable W
2. For $t = 1, \ldots, T$
   - For each example $\mathbf{x}_i, y_i$: *-- do this in random order*
     - Compute the prediction for $\mathbf{x}_i$:

$$p_i = \frac{1}{1 + \exp(-\sum_{j:x_i^j>0} x_i^j w^j)}$$

     - For each non-zero feature of $\mathbf{x_i}$ with index j and value x^j:
       * If j is not in W, set $W[j] = 0$.
       * Set $W[j] = W[j] + \lambda(y_i - p_i)x^j$
3. Output the hash table W.

Learning as optimization for regularized logistic regression

- Algorithm: $w^j = w^j + \lambda(y-p)x^j - \lambda 2\mu w^j$

Time goes from O(nT) to O(mVT) where
- n = number of non-zero entries,
- m = number of examples
- V = number of features
- T = number of passes over data

1. Initialize a hashtable W
2. For $t = 1, \ldots, T$
   - For each example $\mathbf{x}_i, y_i$:
     - Compute the prediction for $\mathbf{x}_i$:

$$p_i = \frac{1}{1 + \exp(-\sum_{j:x_i^j>0} x_i^j w^j)}$$

     - For each ~~non-zero~~ feature of $\mathbf{x_i}$ with index j and value x^j:
       * If j is not in W, set $W[j] = 0$.
       * Set $W[j] = W[j] \boxed{+ \lambda(y-p)x^j - \lambda 2\mu w^j}$
3. Output the hash table W.

This change is very important for large datasets

- We've lost the ability to do *sparse* updates
- This makes learning *much much* more expensive
  - $2*10^6$ examples
  - $2*10^8$ non-zero entries
  - $2*10^6$ + features
  - 10,000x slower (!)

Time goes from O(nT) to O(mVT) where

- n = number of non-zero entries,
- m = number of examples
- V = number of features
- T = number of passes over data

SPARSE UPDATES FOR REGULARIZED LOGISTIC REGRESSION

Learning as optimization for regularized logistic regression

- Final algorithm: $w^j = w^j + \lambda(y - p)x^j - \lambda 2\mu w^j$
- Initialize hashtable *W*
- For each iteration t=1,…T
  - For each example $(\mathbf{x}_i, y_i)$
    - $p_i = \ldots$
    - For each feature *W[j]*
      - $\boxed{W[j] = W[j] - \lambda 2\mu W[j]}$
      - If $x_i^j > 0$ then
        - » $W[j] = W[j] + \lambda(y_i - p^i)x_j$

Learning as optimization for regularized logistic regression

- Final algorithm: $w^j = w^j + \lambda(y - p)x^j - \lambda 2\mu w^j$
- Initialize hashtable *W*
- For each iteration t=1,…T
  - For each example ($\mathbf{x}_i, y_i$)
    - $p_i = \ldots$
    - For each feature $W[j]$
      - $\boxed{W[j] \; *= (1 - \lambda 2\mu)}$
      - If $x_i^j > 0$ then
        - » $W[j] = W[j] + \lambda(y_i - p^i)x_j$

Learning as optimization for regularized logistic regression

- Final algorithm: $w^j = w^j + \lambda(y - p)x^j - \lambda 2\mu w^j$
- Initialize hashtable W
- For each iteration t=1,…T
  - For each example ($\mathbf{x}_i$,y_i)
    - $p_i = \ldots$
    - For each feature $W[j]$
      - If $x_i^j > 0$ then
        - » $\boxed{W[j] \mathrel{*}= (1 - \lambda 2\mu)^A}$
        - » $W[j] = W[j] + \lambda(y_i - p^i)x_j$

A is number of examples seen since the **last** time we did an **x>0 update** on W[j]

Learning as optimization for regularized logistic regression

- Final algorithm: $$w^j = w^j + \lambda(y - p)x^j - \lambda 2\mu w^j$$
- Initialize hashtables W, $\boxed{A}$ and $\boxed{\text{set } k=0}$
- For each iteration t=1,…T
  - For each example $(\mathbf{x}_i, y_i)$
    - $p_i = \ldots$; $\boxed{k{+}{+}}$
    - For each feature $W[j]$
      - If $x_i^j > 0$ then
        - » $\boxed{W[j] \mathrel{*}= (1 - \lambda 2\mu)^{k-A[j]}}$
        - » $W[j] = W[j] + \lambda(y_i - p^i)x_j$
        - » $\boxed{A[j] = k}$

k-A[j] is number of examples seen since the **last** time we did an **x>0 update** on W[j]

Learning as optimization for regularized logistic regression

- Final algorithm: $w^j = w^j + \lambda(y - p)x^j - \lambda 2\mu w^j$
- Initialize hashtables W, $\boxed{A}$ and $\boxed{\text{set } k=0}$
- For each iteration t=1,…T
  - For each example ($\mathbf{x}_i$,y_i)
    - p_i = … ; $\boxed{k{+}{+}}$
    - For each feature $W[j]$
      - If x_i^j>0 then
        - » $\boxed{W[j] \mathrel{*}= (1 - \lambda 2\mu)^{k-A[j]}}$
        - » $W[j] = W[j] + \lambda(y_i - p^i)x_j$
        - » $\boxed{A[j] = k}$

- k = "clock" reading
- A[j] = clock reading last time feature j was "active"
- we implement the "weight decay" update using a "lazy" strategy: weights are decayed in one shot when a feature is "active"

Learning as optimization for regularized logistic regression

- Final algorithm: $w^j = w^j + \lambda(y - p)x^j - \lambda 2 \mu w^j$
- Initialize hashtables W, $\boxed{A}$ and $\boxed{\text{set } k=0}$
- For each iteration t=1,…T
  - For each example ($\mathbf{x}_i$,y_i)
    - p_i = … ; $\boxed{k{+}{+}}$
    - For each feature $W[j]$
      - If x_i^j>0 then
        - » $\boxed{W[j] \mathrel{*}= (1 - \lambda 2\mu)^{k-A[j]}}$
        - » $W[j] = W[j] + \lambda(y_i - p^i)x_j$
        - » $\boxed{A[j] = k}$

Time goes from O(nT) to O(mVT) where

- n = number of non-zero entries,
- m = number of examples
- V = number of features
- T = number of passes over data

Memory use doubles.

Comments

- What's happened here:
  - Our update involves a *sparse part* and a *dense part*
    - Sparse: empirical loss on this example
    - Dense: regularization loss – not affected by the example
  - We remove the *dense part* of the update
    - Old example update:
      - for each feature { do something example-independent}
      - For each active feature { do something example-dependent}
    - New example update:
      - For each active feature :
        - » {simulate the prior example-independent updates}
        - » {do something example-dependent}