Logistic Regression

- For a label-feature pair \((y,x)\), assume the probability model

  \[
p(y|x) = \frac{1}{1 + e^{-yw^Tx}}.
  \]

- \(w\) is the parameter to be decided

- Assume

  \((y_i, x_i), i = 1, \ldots, l\)

  are training instances
Logistic regression finds $\mathbf{w}$ by maximizing the following likelihood

$$\max_{\mathbf{w}} \prod_{i=1}^{l} p(y_i|\mathbf{x}_i). \quad (1)$$

Regularized logistic regression

$$\min_{\mathbf{w}} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^{l} \log \left( 1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i} \right). \quad (2)$$

$C$: regularization parameter decided by users
Gradient-descent Methods

- Given initial $w^0$ and constants $\eta \in (0, 1)$.
- For $k = 0, 1, \ldots$
  - Calculate the direction $s^k = -\nabla f(w^k)$
  - Find $\alpha_k$ satisfying
    \[
    f(w^k + \alpha_k s^k) \leq f(w^k) + \eta \alpha_k \nabla f(w^k)^T s^k
    \]
  - Update $w^{k+1} = w^k + \alpha_k s^k$. 
Gradient

We note that gradient takes the following form

\[ \nabla f(w) = w + C \sum_{i=1}^{l} \left( \frac{e^{-y_i w^T x_i}}{1 + e^{-y_i w^T x_i}} \right) (-y_i x_i) \]

\[ = w + C \sum_{i=1}^{l} \left( \frac{1}{1 + e^{-y_i w^T x_i}} - 1 \right) y_i x_i \]
To find $\alpha_k$ satisfying

$$f(w^k + \alpha_k s^k) \leq f(w^k) + \eta \alpha_k \nabla f(w^k)^T s^k$$

Sequentially check $\alpha_k = 1, 1/2, 1/4, 1/8$

Recall the function is

$$\frac{1}{2} w^T w + C \sum_{i=1}^{l} \log \left( 1 + e^{-y_i w^T x_i} \right).$$

You save time by the property

$$(w + \alpha d)^T x = w^T x + \alpha d^T x$$
You can keep

$$(w^{k+1})^T x = (w^k + \alpha_k s^k)^T x$$

for the next iteration

But error propagation is a concern
Stopping Condition

- You can use
  \[ \| \nabla f(w^k) \| \leq \epsilon \| \nabla f(w^0) \| \]

- This is a relative condition
- You may choose
  \[ \epsilon = 0.01 \]
  Note that a smaller \( \epsilon \) will cause more iterations
- You may need to set a maximal number of iterations as well
Newton direction

\[
\min_s \quad \nabla f(w^k)^T s + \frac{1}{2} s^T \nabla^2 f(w^k) s
\]

\(w^k\): current iterate

This is the same as solving Newton linear system

\[
\nabla^2 f(w^k)s = -\nabla f(w^k)
\]
Given initial $w^0$ and constants $\eta \in (0, 1)$.

For $k = 0, 1, \ldots$

- Solve Newton linear system to obtain direction $s^k$
- Find $\alpha_k$ satisfying
  $$f(w^k + \alpha_k s^k) \leq f(w^k) + \eta \alpha_k \nabla f(w^k)^T s^k$$
- Update $w^{k+1} = w^k + \alpha_k s^k$. 
Newton Linear System

- Hessian $\nabla^2 f(w^k)$ too large to be stored

$$\nabla^2 f(w^k) : n \times n, \quad n : \text{number of features}$$

- But Hessian has a special form

$$\nabla^2 f(w) = \mathcal{I} + C \sum_{i=1}^{l} y_i x_i \left( \frac{e^{-y_i w^T x_i}}{(1 + e^{-y_i w^T x_i})^2} \right) y_i x_i^T$$

$$= \mathcal{I} + CX^T DX$$
Newton Linear System (Cont’d)

- $\mathcal{I}$: identity.

$$X = \begin{bmatrix} x_1^T \\ \vdots \\ x_l^T \end{bmatrix}$$

is the data matrix.

- $D$ diagonal with

$$D_{ii} = \frac{e^{-y_i w^T x_i}}{(1 + e^{-y_i w^T x_i})^2}$$
Using Conjugate Gradient method to solve the linear system.

\[ \nabla^2 f(w^k)s = -\nabla f(w^k) \]

Only a sequence of **Hessian-vector products** are needed

\[ \nabla^2 f(w)s = s + C \cdot X^T(D(Xs)) \]

Therefore, we have a **Hessian-free approach**
Conjugate Gradient

Given $\xi_k < 1$. Let $\bar{s}^0 = 0$, $r^0 = -\nabla f(w^k)$, and $d^0 = r^0$. For $i = 0, 1, \ldots$ (inner iterations)

- If

  $$\|r^i\| \leq \xi_k \|\nabla f(w^k)\|,$$

  then output $s^k = \bar{s}^i$ and stop.

- $\alpha_i = \|r^i\|^2 / ((d^i)^T \nabla^2 f(w^k) d^i)$.

- $\bar{s}^{i+1} = \bar{s}^i + \alpha_i d^i$.

- $r^{i+1} = r^i - \alpha_i \nabla^2 f(w^k) d^i$.

- $\beta_i = \|r^{i+1}\|^2 / \|r^i\|^2$.

- $d^{i+1} = r^{i+1} + \beta_i d^i$. 
The CG stopping condition

\[ \|r^i\| \leq \xi_k \|\nabla f(w^k)\|, \]

is important

It’s a **relative** stopping condition. It becomes strict in the end because of small \( \|\nabla f(w^k)\| \)

Therefore, we **only approximately** obtain the Newton direction
In addition to line search, trust region is another method to ensure sufficient decrease; see the implementation in LIBLINEAR (Lin et al., 2007) http://www.csie.ntu.edu.tw/~cjlin/liblinear

Note that $\alpha_i$ in CG is different from $\alpha_k$ in line search procedure

Check Golub and Van Loan (1996) for details of conjugate gradient methods
Homework

- Implement
  - Gradient-descent method with line search
  - Newton method with line search and CG
  - on MATLAB, Octave, Python, or R
- MATLAB and Octave may be more suitable because of their good support on matrix operations
- Train the data set “kdd2010 (bridge to algebra)” at LIBSVM Data Set http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets
To read data to MATLAB/Octave, check libsvmread.c in the matlab directory of LIBLINEAR.

To be more precise you must build the mex file by

```
>> mex libsvmread.c
```

See liblinear/matlab/README for more details.
Let’s use 

$$\eta = 0.01, \xi_k = 0.1, \mathbf{w}^0 = 0$$

For regularization parameter, set $C = 0.1$

It is known that a larger $C$ causes more iterations

You can check the correctness by comparing with the objective function value of LIBLINEAR (option -s 0 for logistic regression)

You may start with a smaller data set
For Newton method, you should observe that in final iterations, step size $\alpha$ becomes 1. If you don’t see that, you can try to use a smaller $C = 0.01$ and reduce $\epsilon$ to 0.001 or even smaller.

You want to compare gradient-descent and Newton methods.
You may also compare yours with LIBLINEAR. They differ in
  - matlab versus C
  - line search versus trust region to adjust the Newton direction

We require you to submit
  - A report of \( \leq 4 \) pages (without including code)
  - Your code