Logistic Regression

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

$$p(y|\boldsymbol{x}) = \frac{1}{1+e^{-y\boldsymbol{w}^T\boldsymbol{x}}}.$$

- $\boldsymbol{w}$ is the parameter to be decided
- Assume

$$(y_i, \boldsymbol{x}_i), i = 1, \ldots, l$$

are training instances

Logistic Regression (Cont'd)

- Logistic regression finds $\boldsymbol{w}$ by maximizing the following likelihood

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

- Regularized logistic regression

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

C: regularization parameter decided by users

Gradient-descent Methods

- Given initial $\boldsymbol{w}^0$ and constants $\eta \in (0, 1)$.
- For $k = 0, 1, \dots$
  - Calculate the direction $\boldsymbol{s}^k = -\nabla f(\boldsymbol{w}^k)$
  - Find α_k satisfying

$$f(\boldsymbol{w}^k + \alpha_k \boldsymbol{s}^k) \le f(\boldsymbol{w}^k) + \eta \alpha_k \nabla f(\boldsymbol{w}^k)^T \boldsymbol{s}^k$$

  - Update $\boldsymbol{w}^{k+1} = \boldsymbol{w}^k + \alpha_k \boldsymbol{s}^k$.

Gradient

We note that gradient takes the following form

$$
\begin{aligned}
\nabla f(\boldsymbol{w}) &= \boldsymbol{w} + C \sum_{i=1}^{l} \left(\frac{e^{-y_i \boldsymbol{w}^T \boldsymbol{x}_i}}{1 + e^{-y_i \boldsymbol{w}^T \boldsymbol{x}_i}} \right) (-y_i \boldsymbol{x}_i) \\
&= \boldsymbol{w} + C \sum_{i=1}^{l} \left(\frac{1}{1 + e^{-y_i \boldsymbol{w}^T \boldsymbol{x}_i}} - 1 \right) y_i \boldsymbol{x}_i
\end{aligned}
$$

Backtracking Line Search

- To find α_k satisfying

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

- Sequentially check $\alpha_k = 1, 1/2, 1/4, 1/8$
- Recall the function is

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

- You save time by the property

$$(\boldsymbol{w} + \alpha \boldsymbol{d})^T \boldsymbol{x} = \boldsymbol{w}^T \boldsymbol{x} + \alpha \boldsymbol{d}^T \boldsymbol{x}$$

Backtracking Line Search (Cont'd)

- You can keep

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

  for the next iteration
- But error propagation is a concern

Stopping Condition

- You can use

$$\|\nabla f(\boldsymbol{w}^k)\| \leq \epsilon \|\nabla f(\boldsymbol{w}^0)\|$$

- This is a relative condition
- You may choose

$$\epsilon = 0.01$$

  Note that a smaller ϵ will cause more iterations
- You may need to set a maximal number of iterations as well

Newton Methods

- Newton direction

$$\min_{\boldsymbol{s}} \quad \nabla f(\boldsymbol{w}^k)^T \boldsymbol{s} + \frac{1}{2} \boldsymbol{s}^T \nabla^2 f(\boldsymbol{w}^k) \boldsymbol{s}$$

  $\boldsymbol{w}^k$: current iterate
- This is the same as solving Newton linear system

$$\nabla^2 f(\boldsymbol{w}^k) \boldsymbol{s} = -\nabla f(\boldsymbol{w}^k)$$

Newton Methods (Cont'd)

- Given initial $\boldsymbol{w}^0$ and constants $\eta \in (0, 1)$.
- For $k = 0, 1, \ldots$
    - Solve Newton linear system to obtain direction $\boldsymbol{s}^k$
    - Find α_k satisfying

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

    - Update $\boldsymbol{w}^{k+1} = \boldsymbol{w}^k + \alpha_k \boldsymbol{s}^k$.

Newton Linear System

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

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

- But Hessian has a special form

$$\begin{aligned}\nabla^2 f(\boldsymbol{w}) &= \mathcal{I} + C \sum_{i=1}^{l} y_i \boldsymbol{x}_i \left(\frac{e^{-y_i \boldsymbol{w}^T \boldsymbol{x}_i}}{(1 + e^{-y_i \boldsymbol{w}^T \boldsymbol{x}_i})^2} \right) y_i \boldsymbol{x}_i^T \\ &= \mathcal{I} + C X^T D X\end{aligned}$$

Newton Linear System (Cont'd)

- $\mathcal{I}$: identity.

$$X = \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_l^T \end{bmatrix}$$

  is the data matrix.

- D diagonal with

$$D_{ii} = \frac{e^{-y_i \boldsymbol{w}^T \boldsymbol{x}_i}}{(1 + e^{-y_i \boldsymbol{w}^T \boldsymbol{x}_i})^2}$$

Newton Linear System (Cont'd)

- Using Conjugate Gradient method to solve the linear system.

$$\nabla^2 f(\boldsymbol{w}^k)\boldsymbol{s} = -\nabla f(\boldsymbol{w}^k)$$

- Only a sequence of Hessian-vector products are needed

$$\nabla^2 f(\boldsymbol{w})\boldsymbol{s} = \boldsymbol{s} + C \cdot X^T(D(X\boldsymbol{s}))$$

- Therefore, we have a Hessian-free approach

Conjugate Gradient

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

- If
$$\|\boldsymbol{r}^i\| \leq \xi_k \|\nabla f(\boldsymbol{w}^k)\|,$$
then output $\boldsymbol{s}^k = \bar{\boldsymbol{s}}^i$ and stop.
- $\alpha_i = \|\boldsymbol{r}^i\|^2 / ((\boldsymbol{d}^i)^T \nabla^2 f(\boldsymbol{w}^k) \boldsymbol{d}^i)$.
- $\bar{\boldsymbol{s}}^{i+1} = \bar{\boldsymbol{s}}^i + \alpha_i \boldsymbol{d}^i$.
- $\boldsymbol{r}^{i+1} = \boldsymbol{r}^i - \alpha_i \nabla^2 f(\boldsymbol{w}^k) \boldsymbol{d}^i$.
- $\beta_i = \|\boldsymbol{r}^{i+1}\|^2 / \|\boldsymbol{r}^i\|^2$.
- $\boldsymbol{d}^{i+1} = \boldsymbol{r}^{i+1} + \beta_i \boldsymbol{d}^i$.

Conjugate Gradient (Cont'd)

- The CG stopping condition

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

  is important
- It's a relative stopping condition. It becomes strict in the end because of small $\|\nabla f(\boldsymbol{w}^k)\|$
- Therefore, we only approximately obtain the Newton direction

Conjugate Gradient (Cont'd)

- 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 α_i in CG is different from α_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`

Homework (Cont'd)

- 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

Homework (Cont'd)

- Let's use

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

- 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

Homework (Cont'd)

- For Newton method, you should observe that in final iterations, step size α becomes 1.

  If you don't see that, you can try to use a smaller $C = 0.01$ and reduce ϵ to 0.001 or even smaller.
- You want to compare gradient-descent and Newton methods

Homework (Cont'd)

- 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 ≤ 4 pages (without including code)
  - Your code