Case Study 1: Estimating Click Probabilities

Intro
Logistic Regression
Gradient Descent + SGD
AdaGrad

Machine Learning for Big Data
CSE547/STAT548, University of Washington

Emily Fox

January 7th, 2014

Ad Placement Strategies

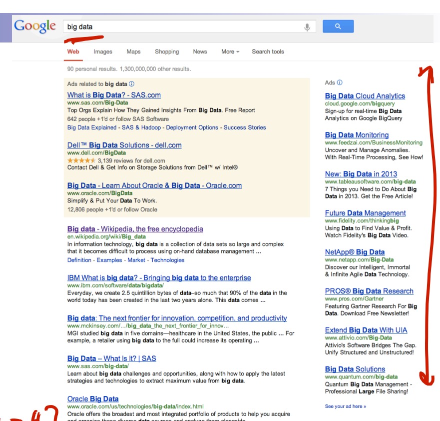
- Companies bid on ad prices

  $c_1 \to \$10$

  $c_2 \to \$20$

  $c_3 \to \$100$

- Which ad wins? (many simplifications here)
  - Naively: $c_3 \to \$100$
  - But: paid on clicks
  - Instead:



e.g. $P(\text{click} \mid c_3) = 0.01 \qquad E[\$_3] = 0.01 \times 100 = \$1$

$P(\text{click} \mid c_2) = 0.5 \qquad E[\$_2] = 0.5 \times 20 = \$10$

Key Task: Estimating Click Probabilities

- What is the probability that user *i* will click on ad *j*
- Not important just for ads:
  - Optimize search results
  - Suggest news articles
  - Recommend products
- Methods much more general, useful for:
  - Classification
  - Regression
  - Density estimation

Learning Problem for Click Prediction

- Prediction task: $X :\to [0,1]$ $\quad P(\text{click}=1 \mid X)$
- Features:

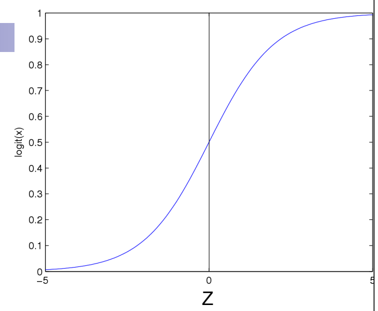
$X = (\text{feats of page}, \text{feats ad}, \text{feats user})$ $\quad y$

$(X, Y) \to (\text{webpage1}, \text{ad 7}, \text{user 25}, \text{time 12}) \to \text{click}=1$

- Data:
  - Batch: fixed dataset $(x^1, y^1) \ldots (x^N, y^N)$
  - Online: data as a stream
    user arrives at a page $\to x^t \to$ predict $\hat{y}$, click? $\to$ observe y^t
- Many approaches (e.g., logistic regression, SVMs, naïve Bayes, decision trees, boosting,…)
  - Focus on logistic regression; captures main concepts, ideas generalize to other approaches

Logistic Regression

Logistic function (or Sigmoid): $\frac{1}{1+exp(-z)}$



- Learn P(Y|**X**) directly
  - Assume a particular functional form
  - Sigmoid applied to a linear function of the data:

$$P(Y=0|X,W) = \frac{1}{1+exp(w_0 + \sum_i w_i X_i)}$$

linear function of features

Features can be discrete or continuous!

Very convenient!

$$P(Y=0\ |X=<X_1,...X_n>) = \frac{1}{1+exp(w_0+\sum_i w_i X_i)}$$

implies

linear classification rule!

$$0 < \ln \frac{P(Y=1\ |X)}{P(Y=0\ |X)} = w_0 + \sum_i w_i X_i > 0 \quad 1$$

predict 1

< 0 predict 0

$w_0+\sum w_i X_i$

1

0

linear decision boundary

Digression: Logistic regression more generally

- Logistic regression in more general case, where *Y in* {y_1,…,y_R}

for *k<R*

$$P(Y = y_k|X) = \frac{\exp(w_{k0} + \sum_{i=1}^n w_{ki}X_i)}{1 + \sum_{j=1}^{R-1} \exp(w_{j0} + \sum_{i=1}^n w_{ji}X_i)}$$

for *k=R* (normalization, so no weights for this class)

$$P(Y = y_R|X) = \frac{1}{1 + \sum_{j=1}^{R-1} \exp(w_{j0} + \sum_{i=1}^n w_{ji}X_i)}$$

Features can be discrete or continuous!

Loss function: Conditional Likelihood

- Have a bunch of iid data of the form:

$$(x^i, y^i)_{1:N} \equiv D = (D_x, D_y)$$

- Discriminative (logistic regression) loss function: **Conditional Data Likelihood**

$$\arg\max_w P(D_y | D_x, w) \overset{iid}{\equiv} \arg\max_w \prod_{j=1}^N P(y^j | x^j, w)$$

$$\equiv \arg\max_w \ln \prod_{j=1}^N P(y^j | x^j, w)$$

$$\underbrace{\ln P(\mathcal{D}_Y \mid \mathcal{D}_\mathbf{X}, \mathbf{w}) = \sum_{j=1}^N \ln P(y^j \mid \mathbf{x}^j, \mathbf{w})}$$

goal to max this

Expressing Conditional Log Likelihood

$$P(Y=0|\mathbf{X},\mathbf{w}) = \frac{1}{1+exp(w_0+\sum_i w_i X_i)}$$

$$P(Y=1|\mathbf{X},\mathbf{w}) = \frac{exp(w_0+\sum_i w_i X_i)}{1+exp(w_0+\sum_i w_i X_i)}$$

argmax w $l(\mathbf{w}) \equiv \sum_j \ln P(\overset{\{0,1\}}{y^j}|\mathbf{x}^j,\mathbf{w})$

$$= \sum_j \begin{cases} P(Y=1|x^j,w), & \text{if } y^j=1 \\ P(Y=0|x^j,w), & \text{if } y^j=0 \end{cases}$$

$$\ell(\mathbf{w}) = \sum_j y^j \ln P(Y=1|\mathbf{x}^j,\mathbf{w}) + (1-y^j)\ln P(Y=0|\mathbf{x}^j,\mathbf{w})$$

$$= \sum_j y^j(w_0 + \sum_{i=1}^d w_i x_i^j) - \ln\left(1+\exp(w_0+\sum_{i=1}^d w_i x_i^j)\right)$$

Argmax w

for LR

Maximizing Conditional Log Likelihood

$$l(\mathbf{w}) \equiv \ln \prod_j P(y^j|\mathbf{x}^j,\mathbf{w})$$

$$= \sum_j y^j(w_0 + \sum_{i=1}^d w_i x_i^j) - \ln\left(1+\exp(w_0+\sum_{i=1}^d w_i x_i^j)\right)$$

Good news: *l*(**w**) is concave function of **w**, no local optima problems

Bad news: no closed-form solution to maximize *l*(**w**)

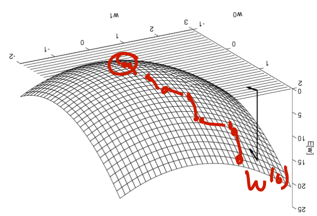
Good news: concave functions easy to optimize

Optimizing concave function – Gradient ascent

– f is concave if −f is convex

diff. scalar fns: convex $f''(x) \geq 0 \ \forall x$

- Conditional likelihood for Logistic Regression is concave. Find optimum with gradient ascent



Gradient: $$\nabla_{\mathbf{w}} l(\mathbf{w}) = [\frac{\partial l(\mathbf{w})}{\partial w_0}, \ldots, \frac{\partial l(\mathbf{w})}{\partial w_n}]'$$

Step size, η>0

Update rule: $$\Delta \mathbf{w} = \eta \nabla_{\mathbf{w}} l(\mathbf{w})$$

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \frac{\partial l(\mathbf{w})}{\partial w_i}$$

- Gradient ascent is simplest of optimization approaches
  - e.g., Conjugate gradient ascent much better (see reading)

Gradient Ascent for LR

Gradient ascent algorithm: iterate until change < ε

$$w_0^{(t+1)} \leftarrow w_0^{(t)} + \eta \sum_j \overbrace{[y^j - \hat{P}(Y^j = 1 \mid \mathbf{x}^j, \mathbf{w}^{(t)})]}^{\text{truth}}$$

prediction

weigh by feature value

For i = 1,…,d,

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \sum_j x_i^j [y^j - \hat{P}(Y^j = 1 \mid \mathbf{x}^j, \mathbf{w}^{(t)})]$$

repeat

Regularized Conditional Log Likelihood

- If data is linearly separable, weights go to infinity
- Leads to overfitting → Penalize large weights $\sum_{i=1}^{d} w_i^2$
- Add regularization penalty, e.g., L_2:

$$\underset{w}{\text{argmax}} \quad \ell(\mathbf{w}) = \underbrace{\ln \prod_j P(y^j|\mathbf{x}^j,\mathbf{w}))}_{\text{conditional likelihood}} - \underbrace{\frac{\lambda}{2}||\mathbf{w}||_2^2}_{\text{"regularization" penalty}}$$

- Practical note about w_0:

Don't regularize w_0...

Standard v. Regularized Updates

- Maximum conditional likelihood estimate

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \ln \left[\prod_{j=1}^{N} P(y^j \mid \mathbf{x}^j, \mathbf{w}) \right]$$

$$\boxed{w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \sum_j x_i^j [y^j - \hat{P}(Y^j = 1 \mid \mathbf{x}^j, \mathbf{w}^{(t)})]}$$

- Regularized maximum conditional likelihood estimate

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \ln \left[\prod_j P(y^j|\mathbf{x}^j,\mathbf{w})) \right] - \frac{\lambda}{2} \sum_{i>0} w_i^2$$

$$\boxed{w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \left\{ \underbrace{-\lambda w_i^{(t)}}_{\text{reg derivative} \leftarrow \text{move towards zero}} + \sum_j x_i^j [y^j - \hat{P}(Y^j = 1 \mid \mathbf{x}^j, \mathbf{w}^{(t)})] \right\}}$$

Stopping criterion

for LR ✓ strongly concave with $\gamma \geq \lambda$

$$\ell(\mathbf{w}) = \ln \prod_j P(y^j|\mathbf{x}^j, \mathbf{w})) - \frac{\lambda}{2}||\mathbf{w}||_2^2$$

- Regularized logistic regression is strongly concave
  - Negative second derivative bounded away from zero:

$f(x)$

concave diff $\iff -f''(x) \geq 0$; strongly concave : $-f''(x) \geq \gamma$, $\gamma > 0$

- Strong concavity (convexity) is super helpful!!

stop $\ell(w^*) - \ell(w^{(t)}) \leq \varepsilon$

- For example, for strongly concave $l(\mathbf{w})$:

$$\underbrace{\ell(\mathbf{w}^*)}_{\text{OPT}} - \underbrace{\ell(\mathbf{w})}_{\text{value at some iter}} \leq \frac{1}{2\lambda}||\nabla \ell(\mathbf{w})||_2^2 \leq \varepsilon \implies ||\nabla \ell(w^{(t)})||_2^2 \leq 2\lambda\varepsilon$$

Convergence rates for gradient descent/ascent

- Number of Iterations to get to accuracy

$$\ell(\mathbf{w}^*) - \ell(\mathbf{w}) \leq \epsilon$$

ℓ

- If func Lipschitz: O(1/ϵ²)
  $||\ell(u) - \ell(v)|| \leq K ||u-v||$
- If gradient of func Lipschitz: O(1/ϵ)
  $||\nabla \ell(u) - \nabla \ell(v)|| \leq K||u-v||$
- If func is strongly convex: O(ln(1/ϵ))

regularized LR

exp faster

constant step size (sufficiently small)

Challenge 1: Complexity of computing gradients

d features

- What's the cost of a gradient update step for LR???

cache

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \left\{ -\lambda w_i^{(t)} + \underbrace{\sum_{j=1}^{N} x_i^j [y^j - \underbrace{\widehat{P}(Y^j = 1 \mid \mathbf{x}^j, \mathbf{w}^{(t)})}_{O(d)}]}_{O(Nd)} \right\}$$

for each i

$\forall$ features i, cost is $O(Nd^2)$; cache $P(y^j=1 \mid x^j, w^{(t)})$ $O(Nd)$

in Big Data N is very large:

$O(Nd) \rightarrow$ only taking little η step

Challenge 2: Data is streaming

- Assumption thus far: **Batch data** $\sum_{j=1}^{N} \ldots\ldots$
- But, click prediction is a streaming data task:
  - User enters query, and ad must be selected:
    - Observe $\mathbf{x}^j$, and must predict y^j

$\rightarrow$ [] $\rightarrow$ x^j $\rightarrow$ predict $\hat{y}^j$ $\rightarrow$ show ad

  - User either clicks or doesn't click on ad:
    - Label y^j is revealed afterwards
      - Google gets a reward if user clicks on ad
  - Weights must be updated for next time:

$w^{(t+1)} \leftarrow w^{(t)} + \Delta$

Learning Problems as Expectations

- Minimizing loss in training data:
  - Given dataset: $x^1 \ldots x^N$
    - Sampled iid from some distribution p(**x**) on features: $x^j \overset{iid}{\sim} p(x)$
  - Loss function, e.g., hinge loss, logistic loss,…
  - We often minimize loss in training data:

$$\ell_{\mathcal{D}}(\mathbf{w}) = \underbrace{\frac{1}{N}\sum_{j=1}^{N}\ell(\mathbf{w},\mathbf{x}^j)}$$

monte carlo integration

- However, we should really minimize expected loss on all data:

$$\underbrace{\ell(\mathbf{w})}_{\text{true}} = \underbrace{E_{\mathbf{x}}\left[\ell(\mathbf{w},\mathbf{x})\right]}_{\text{expected loss}} = \int p(\mathbf{x})\ell(\mathbf{w},\mathbf{x})d\mathbf{x}$$

- So, we are approximating the integral by the average on the training data

$\nabla \Sigma = \Sigma \nabla$

Gradient ascent in Terms of Expectations

- "True" objective function:

$$\ell(\mathbf{w}) = E_{\mathbf{x}}\left[\ell(\mathbf{w},\mathbf{x})\right] = \int p(\mathbf{x})\ell(\mathbf{w},\mathbf{x})d\mathbf{x}$$

- Taking the gradient:

$$\nabla_w \ell(w) = \nabla_w E_x[\ell(w,x)] = E_x[\nabla_w \ell(w,x)]$$

- "True" gradient ascent rule:

$$w^{(t+1)} \leftarrow w^{(t)} + \eta \underbrace{E_x[\nabla_w \ell(w,x)]}$$

estimate exp. gradient from data

- How do we estimate expected gradient?

if data on machine, must shuffle first

SGD: Stochastic Gradient Ascent (or Descent)

- "True" gradient: $\nabla \ell(\mathbf{w}) = E_{\mathbf{x}}[\nabla \ell(\mathbf{w}, \mathbf{x})]$

$x \overset{iid}{\sim} p(x)$

- Sample based approximation:

$$\nabla \ell(w) = E_x[\nabla \ell(w,x)] \approx \hat{\nabla} \ell(w) = \frac{1}{N}\sum_{j=1}^{N} \nabla \ell(w,x)$$

the bigger N, the closer $\hat{\nabla}\ell$ is $\nabla \ell$

- What if we estimate gradient with just one sample???
  - Unbiased estimate of gradient $\nabla \ell(w) \approx \hat{\nabla}\ell(w) = \nabla \ell(w, x^{(t)})$
  - Very noisy! $E[\hat{\nabla}\ell(w)] = E_{x^{(t)}}[\nabla \ell(w, x^{(t)})] = \nabla \ell(w)$
  - Called stochastic gradient ascent (or descent)
    - Among many other names
  - VERY useful in practice!!!

Stochastic Gradient Ascent: general case

running avg of $w^{(t)}$ is used for prediction

- Given a stochastic function of parameters: $f(w) = E_x[f(w,x)]$
  - Want to find maximum

$w^* \in \arg\min_w E_x[f(w,x)]$

- Start from $\mathbf{w}^{(0)}$ e.g., $w^{(0)} = 0$
- Repeat until convergence:
  - Get a sample data point $\mathbf{x}^t$ ↳ sell ad, observe $y^t \in$ click?
  - Update parameters:

$$w^{(t+1)} \leftarrow w^{(t)} + \eta_t \nabla f(w^{(t)}, x^t)$$

actual params, not avg

Stopping criteria?
good luck...
in online: never stop
w/ batch alg:
- validation set
- progressive validation

- Works on the online learning setting!
- Complexity of each gradient step is constant in number of examples!
- In general, step size changes with iterations

e.g. $\eta_t = K/t$ for $K > 0$

Stochastic Gradient Ascent for Logistic Regression

- Logistic loss as a stochastic function:

$$E_{\mathbf{x}}\left[\ell(\mathbf{w},\mathbf{x})\right] = E_{\mathbf{x}}\left[\ln P(y|\mathbf{x},\mathbf{w}) - \frac{\lambda}{2}||\mathbf{w}||_2^2\right]$$

- Batch gradient ascent updates: O(Nd)

$$\boxed{w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta\left\{-\lambda w_i^{(t)} + \frac{1}{N}\sum_{j=1}^{N} x_i^{(j)}[y^{(j)} - P(Y=1|\mathbf{x}^{(j)},\mathbf{w}^{(t)})]\right\}}$$

avg all datapoint

- Stochastic gradient ascent updates: O(d)
  - Online setting:

$$\boxed{w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta_t\left\{-\lambda w_i^{(t)} + x_i^{(t)}[y^{(t)} - P(Y=1|\mathbf{x}^{(t)},\mathbf{w}^{(t)})]\right\}}$$

1 data point at a time