Case Study 1: Estimating Click Probabilities

Intro
Logistic Regression
Gradient Descent + SGD

Machine Learning for Big Data
CSE547/STAT548, University of Washington
Emily Fox
March 31, 2015

Ad Placement Strategies

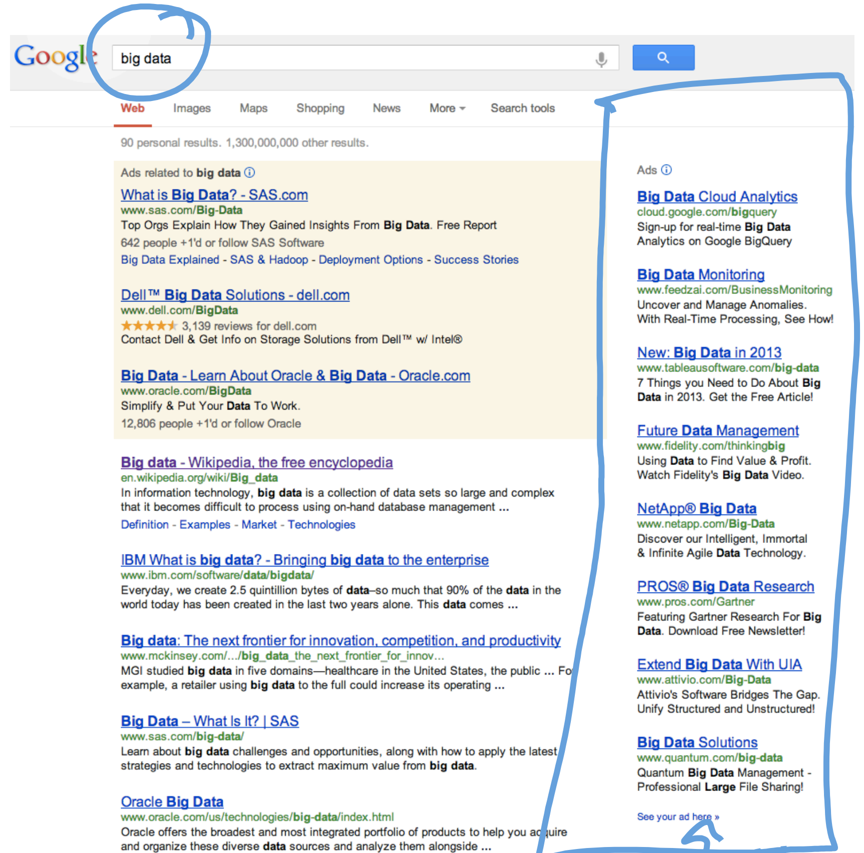
- Companies bid on ad prices

  $c_1 \rightarrow \$10$
  $c_2 \rightarrow \$20$
  $c_3 \rightarrow \$100$

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

e.g. $P(\text{click} \mid c_3) = 0.01$ *
$P(\text{click} \mid c_2) = 0.5$

$E[\$_3] = 0.01 \times 100 = \$1$
$E[\$_2] = 0.5 \times 20 = \$10 \leftarrow$ wins



ads

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

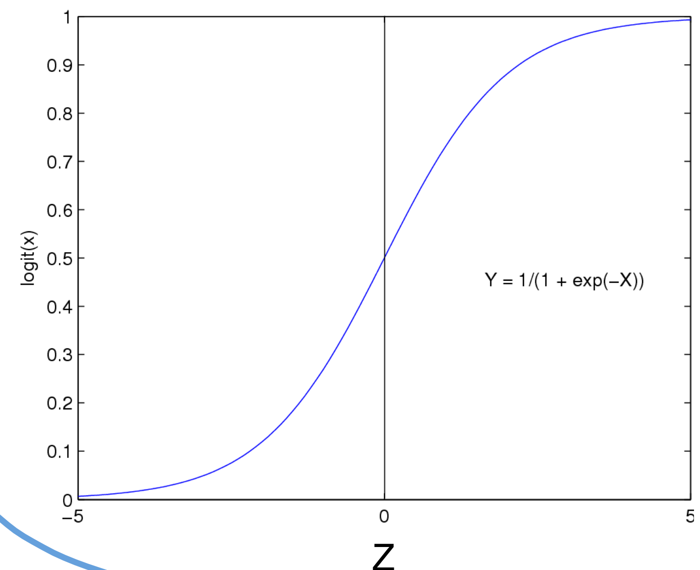
- Prediction task: $X \to \{0,1\}$ $\quad P(\text{click}=1 \mid X)$
- Features: $X = (\text{feats of page, ad, user})$
- Data: (x^i, y^i) $\quad (\text{webpage1, ad7, user25, time12}) \leftarrow x^i$
  $\text{click}=1 \leftarrow y^i$
  - Batch: Fixed dataset $(x^1, y^1) \ldots (x^N, y^N)$
  - Online: data as a stream
    user arrives at a page $\to X^t$ $\nearrow$ predict $\hat{y}$ click? $\searrow$ 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

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

$$P(Y=0|X,W) = \frac{1}{1+exp(\underbrace{w_0+\sum_i w_i X_i}_{\text{linear fcn of features } = z})}$$

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



Features can be discrete or continuous!

Very convenient!

Note: $\sum w_i x_i = \langle w, X \rangle$

$$P(\underline{Y=0} \mid X = \langle X_1, \dots X_n \rangle) = \frac{1}{1 + exp(w_0 + \sum_i w_i X_i)}$$

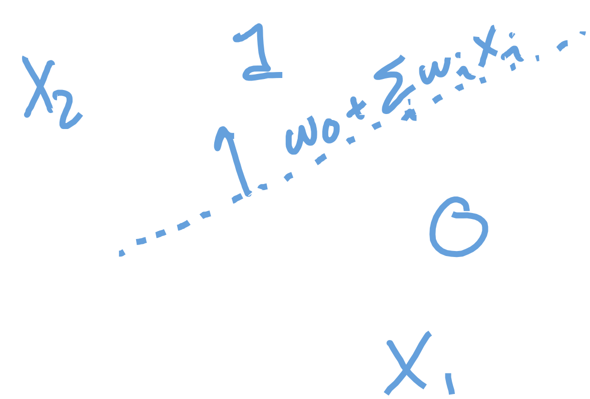
linear classification rule!

implies

$$\ln \frac{P(Y=1 \mid X)}{P(Y=0 \mid X)} = w_0 + \sum_i w_i X_i \quad \begin{matrix} \text{predict 1} \\ \gtrless 0 \\ \text{predict 0} \end{matrix}$$

"log odds"

2d example



Digression: Logistic regression more generally

- Logistic regression in more general case, where *Y* in $\{y_1,\ldots,y_R\}$

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:

$$(x^i, y^i)_{1:N} \triangleq \mathcal{D} = (\mathcal{D}_x, \mathcal{D}_y)$$

- Discriminative (logistic regression) loss function:

  Conditional Data Likelihood

$$\arg\max_w P(\mathcal{D}_y \mid \mathcal{D}_x, w) \overset{iid}{=} \arg\max_w \prod_{j=1}^{N} P(y^j \mid x^j, w)$$

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

max this:

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

Expressing Conditional Log Likelihood

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

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

$\{0,1\}$

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

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

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

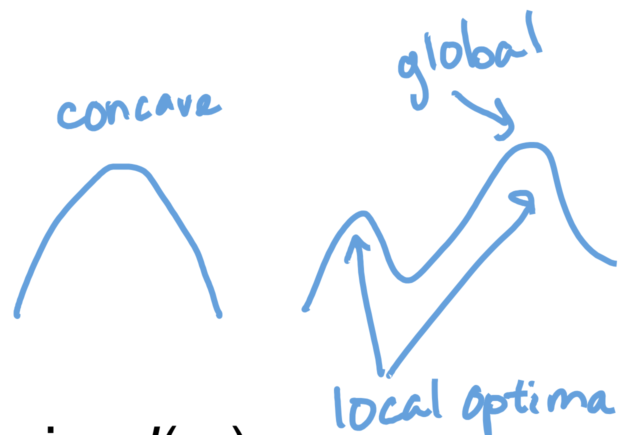
$$= \sum_j y^j(w_0 + \sum_{i=1}^d w_i x_i^j) - \ln\left(1+\exp(w_0+\sum_{i=1}^d w_i x_i^j)\right)$$

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 using iterative methods

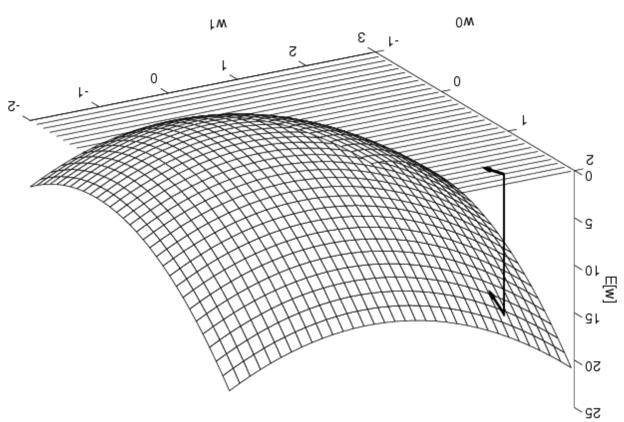
Optimizing concave function – Gradient ascent

f concave
$\Updownarrow$
$-f$ convex

- Conditional likelihood for logistic regression is *concave*
- Find optimum with *gradient ascent*

g convex

$g''(x) \geq 0 \;\; \forall x$



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

truth

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

For $i = 1,\ldots,d$,

weigh by feature value

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

repeat

Regularized Conditional Log Likelihood

- If data are linearly separable, weights go to infinity
- Leads to overfitting → Penalize large weights

- Add regularization penalty, e.g., L_2:

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

$\sum_{i=1}^{d} w_i^2$

- Practical note about w_0:

don't regularize w_0

(redefine w)

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

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

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

neg. derivative ← move towards 0

Stopping criterion

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

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

Stop if $\ell(w^*) - \ell(w^{(t)}) \leq \epsilon$

$\Downarrow$

$||\nabla \ell(w^{(t)})||_2^2 \leq 2\lambda\epsilon$

computable

Convergence rates for gradient descent/ascent

- Number of iterations to get to accuracy

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

- If func ℓ Lipschitz: $O(1/\epsilon^2)$

$$\| \ell(u) - \ell(v) \| \leq k \| u - v \|$$

- If gradient of func Lipschitz: $O(1/\epsilon)$

$$\| \nabla \ell(u) - \nabla \ell(v) \| \leq k \| u - v \|$$

constant step size (sufficiently small)

- If func is strongly convex: $O(\ln(1/\epsilon))$

e.g. reg. LR

exponentially fewer iterations

Challenge 1: Complexity of computing gradients

d features

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

cache

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

for each i

$$\mathbf{w}^{(t)} = \begin{bmatrix} w_1^{(t)} \\ \vdots \\ w_d^{(t)} \end{bmatrix}$$

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

$\downarrow$

$O(Nd)$

In "big data" - N is very large

$O(Nd)$ for only taking little η step

Challenge 2: Data is streaming

- Assumption thus far: **Batch data**

$$\sum_{j=1}^{N} \ldots$$

- 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 $\longrightarrow$ predict $\hat{y}^j$ $\longrightarrow$ show ad

click?

  - 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:

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

depends just on recent example(s)

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: $x^i \overset{iid}{\sim} p(x)$
  - Loss function, e.g., hinge loss, logistic loss,...
  - We often minimize loss in training data:

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

Monte Carlo integration w/ N samples

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

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

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

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:

$$w^{(t+1)} \leftarrow w^{(t)} + \eta \underbrace{E_x[\nabla_w \ell(w,x)]}_{\text{estimate expected gradient from data}}$$

- How do we estimate expected gradient?

SGD: Stochastic Gradient Ascent (or Descent)

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

- Sample based approximation: $x^j \overset{iid}{\sim} p(x)$

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

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

- What if we estimate gradient with just one sample??? $N=1$, $x^{(t)}$
  - Unbiased estimate of gradient
  - Very noisy!
  - Called stochastic gradient ascent (or descent)
    - Among many other names
  - VERY useful in practice!!!

$$\nabla \ell(w) \approx \hat{\nabla} \ell(w) = \nabla \ell(w, x^{(t)})$$

$$E_x[\hat{\nabla} \ell(w)] = E_{x^{(t)}}[\nabla \ell(w, x^{(t)})] = \nabla \ell(w)$$