Generalized Linear Models

CS 229: Machine Learning
Sanmi Koyejo
Stanford University
(adapted from slides by Chris Ré; Emily Fox)

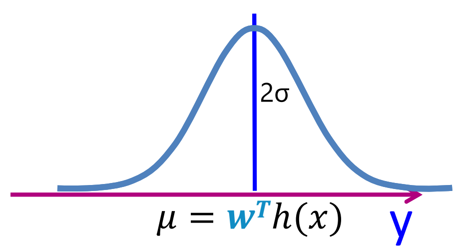
1

Models for data; how are these related?

- Linear regression with Gaussian errors

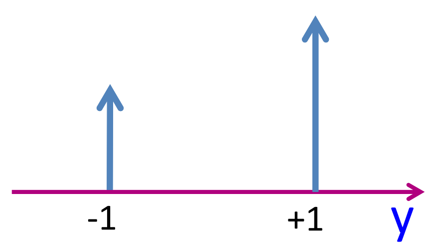
  $y_i = \boldsymbol{w}^{\boldsymbol{T}} h(x_i) + \ \varepsilon_i \ ; \ \varepsilon_i \sim N(0, \sigma^2)$

  ➔ $p(y|x, \boldsymbol{w}) = N(y; \boldsymbol{w}^{\boldsymbol{T}} h(x), \sigma^2)$



- Logistic regression

$$P(y = +1|x, w) = \frac{1}{1+\exp(-\boldsymbol{w}^{\boldsymbol{T}} h(x))}$$



2

Background: Exponential Family Models

- **Rough motivation:** If P has a special form, then we can standardize (simplify) inference and learning

$$P(y;\eta) = b(y)\exp\left\{\eta^T T(y) - a(\eta)\right\}$$

- Here y , $a(\eta)$, and $b(y)$ are scalars. $T(y)$ is same dimension as the η.

These terms have names:

- η is called the natural parameter (also called canonical parameter)
- $T(y)$ is called the sufficient statistic.
- $b(y)$ is called the base measure, does not depend on η.
- $a(\eta)$ is called the log partition function, does not depend on y.

Example: Bernoulli $P(y;\phi) = \phi^y\,(1-\phi)^{1-y}$

- A Bernoulli random variable is an event (e.g., flipping a coin)
- How do we write this in exponential family form?

$$P(y;\eta) = b(y)\exp\left\{\eta^T T(y) - a(\eta)\right\} \quad e^{\log(\phi^y(1-\phi)^{1-y})}$$

- $\phi^y\,(1-\phi)^{1-y}$
$$= \exp\{y\log\phi + (1-y)\log(1-\phi)\}$$
$$= \exp\left\{y\log\frac{\phi}{1-\phi} + \log(1-\phi)\right\} \quad -\log(1+e)$$

- So $\quad \boxed{\eta = \log\frac{\phi}{1-\phi}}, T(y) = y, a(\eta) = -\log(1-\phi).$

Example: Gaussian, with fixed variance $\sigma^2 = 1$

$$P(y; \mu) = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{1}{2}(y-\mu)^2\right\}.$$

- How do we write this in exponential family form?

$$P(y; \eta) = b(y) \exp\left\{\eta^T T(y) - a(\eta)\right\}$$

- Multiply out the square and group the terms

$$P(y; \mu) = \frac{1}{\sqrt{2\pi}} \exp\left\{-y^2/2\right\} \exp\left\{\mu y - \frac{1}{2}\mu^2\right\}.$$

- So

$$\eta = \mu,\ T(y) = y,\ a(\eta) = \frac{1}{2}\eta^2.$$

CS 229: Machine Learning

5

The log partition function; $a(\eta)$ ensures normalization

$$P(y; \eta) = b(y) \exp\left\{\eta^T T(y) - a(\eta)\right\}$$

$$1 = \sum_y P(y; \eta) = e^{-a(\eta)} \sum_y b(y) \exp\left\{\eta^T T(y)\right\} = 1$$

$$\implies a(\eta) = \log \sum_y b(y) \exp\left\{\eta^T T(y)\right\}$$

Can compute all moments from gradients of $a(\eta)$; see notes Ch 3

CS 229: Machine Learning

6

There are many canonical exponential family models!

$$P(y;\eta) = b(y)\exp\left\{\eta^T T(y) - a(\eta)\right\}$$

- Binary: Bernoulli
- Multiple classes: Multinomial (equiv. categorical)
- Real-valued: Gaussian
- Counts: Poisson
- Positive Reals: Gamma, Exponential
- Distributions: Dirichlet

For this course, we will generally use models where $T(y) = y$

Expectation for exponential family models

$$P(y;\eta) = b(y)\exp\left\{\eta^T T(y) - a(\eta)\right\}$$

- $E[T(y);\ \eta] = g(\eta)$; g is called the canonical response function
  - **NOTE:** g^{-1} is the canonical link function; notation varies (see class notes, Ch 3)

	Mean	response function g
Gaussian	μ	identity function
Bernoulli (assuming y in {0,1} instead of {-1,1} or some other set of values)	P(y=+1) (mean is different if y is not in {0,1})	sigmoid function (need slight modification if y is not in {0,1})

Generalized Linear Models (GLMs)
using Exponential Family Models

Recipe for Generalized linear Models

- Given input x and target y
- First: Pick a distribution based on y's type
  - Binary: Bernoulli
  - Multiple classes: Multinomial (equiv. categorical)
  - Real-valued: Gaussian
  - Counts: Poisson
  - Positive Reals: Gamma, Exponential
  - Distributions: Dirichlet

Recipe for Generalized Linear Models (GLMs)

Our model is linear in the natural parameters $\eta(x) = w^T h(x)$ and $T(y) = y$

- **Inference:** $\mu(x) = E[y|x; w] = g(w^T h(x))$ is the output
- **Learning:** $\max_w \log p(y|x; w)$, i.e., maximum likelihood
- **Algorithm:** (stochastic) gradient update (more on this later)

$$w^{\{t+1\}} = w^t + \alpha\left(y^i - \mu(x^i)\right)h(x^i)$$

Examples of GLMs

When $T(y) = y$, a generalized linear model has the property

$$E[y|x, \boldsymbol{w}] = g(\boldsymbol{w}^T h(x))$$

	Mean	Response function g
Linear regression	$\boldsymbol{w}^T h(x)$	identity function
Logistic regression (assuming y in {0,1} instead of {-1,1} or some other set of values)	$P(y = +1 \| x, \boldsymbol{w})$ (mean is different if y is not in {0,1})	sigmoid function (need slight modification if y is not in {0,1})

Similarly with multinomial, Poisson, gamma, exponential, …

Multiclass Classification
(aka softmax regression)

CS 229: Machine Learning

13

Suppose we want to choose among k discrete values, e.g., {'Cat', 'Dog', 'Car', 'Bus'}, so $k = 4$

Encode as one-hot vectors, i.e., $y \in \{0,1\}^k$ and $\sum_j y_j = 1$

$P(y=i) = \theta^i$

$P(y) = \prod_{i=1}^{k} [\theta^i]^{1[y=i]}$

$$\begin{array}{cccc} \begin{pmatrix}1\\0\\0\\0\end{pmatrix} & \begin{pmatrix}0\\1\\0\\0\end{pmatrix} & \begin{pmatrix}0\\0\\1\\0\end{pmatrix} & \begin{pmatrix}0\\0\\0\\1\end{pmatrix} \\ \text{'Cat'} & \text{'Dog'} & \text{'Car'} & \text{'Bus'} \end{array}$$

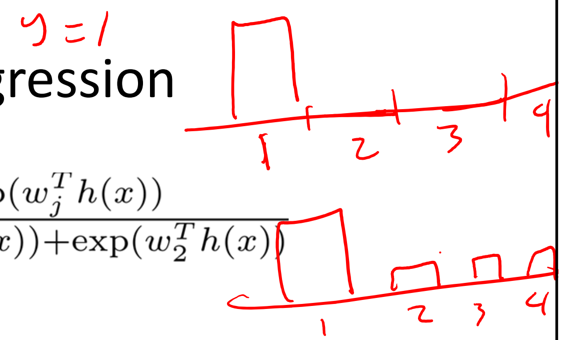
A prediction is a distribution over k classes. We represent this using the SoftMax function (see notes Ch3 for derivation)

$$P(y=j|x;w) = \frac{\boxed{\exp(w_j^T h(x))}}{\sum_{i=1}^{k} \exp(w_i^T h(x))} \propto \theta^j$$

CS 229: Machine Learning

14

Compare k=2 to logistic regression



$$P(y = j|x; w) = \frac{\exp(w_j^T h(x))}{\exp(w_1^T h(x)) + \exp(w_2^T h(x))}$$

Hint, compare two class softmax regression vs. logistic regression using parameter $w_1 - w_2$

- **Note:** For general k, can write an equivalent model with k-1 classes (since probability must sum to 1)
- **Also note:** Multinomials are in the exponential family (see class notes Ch 3), can run everything learned so far!

Training multiclass classification (direct approach)

Let

$$\hat{p}_j = P(y = j|x; w) = \frac{\exp(w_j^T h(x))}{\sum_{i=1}^{k} \exp(w_i^T h(x))}$$

Maximize the probability of the given class! Popularly known as:

CROSSENTROPY$(p, \hat{p}) = \sum_j p(y = j|x) \log \hat{p}(y = j|x)$

p is the label (one hot vector), so this reduces to

$$-\log \hat{p}(y = i|x) = -\log \frac{\exp(w_i^T h(x))}{\sum_{j=1}^{k} \exp(w_j^T h(x))}$$

What you can do now…

- recognize exponential family models, and some of their properties
- recognize how exponential family models are related to generalized linear models
- recognize some special cases of exponential family models for regression, (binary, multiclass) classification, count regression
- inference and learning for generalized linear models
- familiar with SOFTMAX and CROSSENTROPY for multiclass classification

CS 229: Machine Learning

17

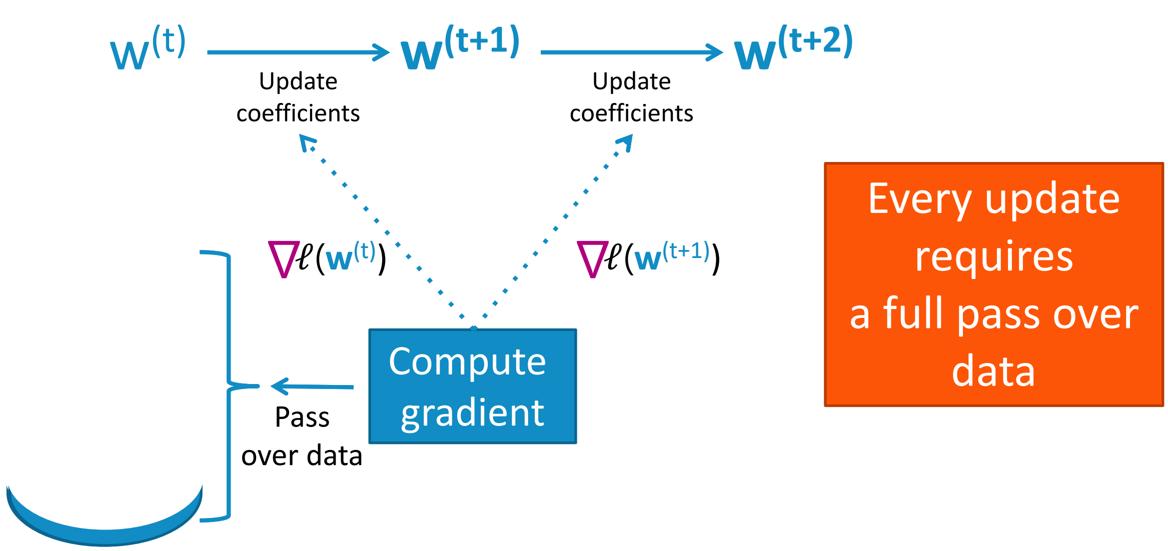
Scaling up learning via SGD

CS 229: Machine Learning
Sanmi Koyejo
Stanford University
(adapted from slides by Emily fox and Carlos Guestrin)

18

Stochastic gradient ascent/descent: Learning, one data point at a time

Why gradient ascent is slow…



More formally: How expensive is gradient ascent?

Sum over data points

$$\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_j} = \sum_{i=1}^{N} \underbrace{h_j(\mathbf{x}_i)\Big(\mathbb{1}[y_i = +1] - P(y = +1 \mid \mathbf{x}_i, \mathbf{w})\Big)}_{\text{Contribution of data point } x_i,y_i \text{ to gradient}}$$

Contribution of data point x_i,y_i to gradient

$$\frac{\partial \ell_i(\mathbf{w})}{\partial \mathbf{w}_j}$$

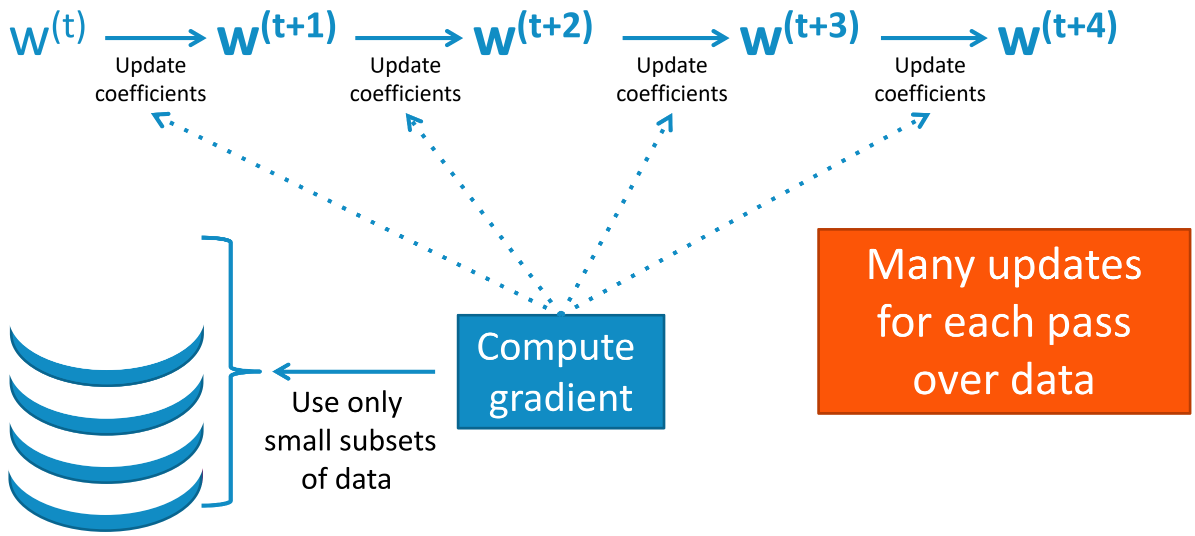
Every step requires touching every data point!!!

Sum over data points

$$\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_j} = \sum_{i=1}^{N} \frac{\partial \ell_i(\mathbf{w})}{\partial \mathbf{w}_j}$$

Time to compute contribution of $\mathbf{x}_i,\mathbf{y}_i$	# of data points (N)	Total time to compute 1 step of gradient ascent
1 millisecond	1000	1 sec
1 second	1000	16.7 mins
1 millisecond	10 million	2.8 hrs
1 millisecond	10 billion	115.7 days

Stochastic gradient ascent



23

Example: Instead of all data points for gradient, use 1 data point only???

Sum over data points

Gradient ascent

$$\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_j} = \sum_{i=1}^{N} \frac{\partial \ell_i(\mathbf{w})}{\partial \mathbf{w}_j}$$

Stochastic gradient ascent

Each time, pick different data point i

$$\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_j} \approx \frac{\partial \ell_i(\mathbf{w})}{\partial \mathbf{w}_j}$$

24

Stochastic gradient ascent for logistic regression

init $w^{(1)}$=0, t=1

until converged

for i=1,...,N

for j=0,...,D

Sum over data points

Each time, pick different data point i

partial[j] = $\cancel{\sum_{i=1}^{N}} h_j(\mathbf{x}_i)\Big(\mathbb{1}[y_i = +1] - P(y = +1 \mid \mathbf{x}_i, \mathbf{w}^{(t)})\Big)$

$w_j^{(t+1)} \leftarrow w_j^{(t)} + \eta$ partial[j]

t ← t + 1

25

Stochastic gradient for L2-regularized objective

Total derivative = $\sum_{i=1}^{N} \frac{\partial \ell_i(\mathbf{w})}{\partial \mathbf{w}_j} - 2\,\lambda\, w_j$ $\boxed{\text{partial[j]}}$

What about regularization term?

Each time, pick different data point i

Stochastic gradient ascent

Total derivative $\approx \frac{\partial \ell_i(\mathbf{w})}{\partial \mathbf{w}_j} - \frac{2}{N}\,\lambda\, w_j$ $\boxed{\text{partial[j]}}$

Each data point contributes 1/N to regularization

26

Comparing computational time per step

Gradient ascent

$$\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_j} = \sum_{i=1}^{N} \frac{\partial \ell_i(\mathbf{w})}{\partial \mathbf{w}_j}$$

Stochastic gradient ascent

$$\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_j} \approx \frac{\partial \ell_i(\mathbf{w})}{\partial \mathbf{w}_j}$$

Time to compute contribution of $\mathbf{x}_i,\mathbf{y}_i$	# of data points (N)	Total time for 1 step of gradient	Total time for 1 step of stochastic gradient
1 millisecond	1000	1 second	1 ms
1 second	1000	16.7 minutes	1 s
1 millisecond	10 million	2.8 hours	1 ms
1 millisecond	10 billion	115.7 days	1 ms

27

Which one is better??? Depends…

Algorithm	Time per iteration	Total time to convergence for large data: In theory	Total time to convergence for large data: In practice	Sensitivity to parameters
Gradient	Slow for large data	Slower	Often slower	Moderate
Stochastic gradient	Always fast	Faster	Often faster	Very high

28

Summary of stochastic gradient

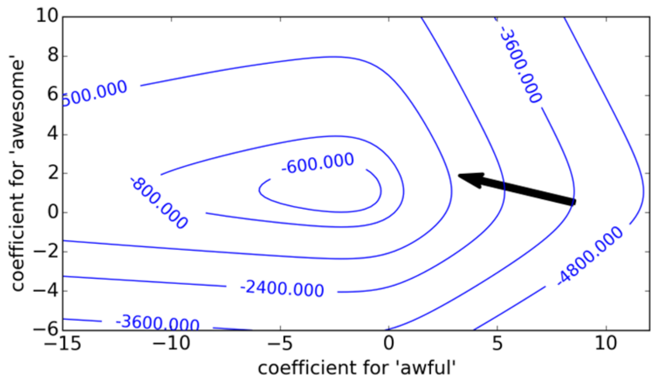
- Tiny change to gradient ascent
- Much better scalability
- Huge impact in real-world
- Very tricky to get right in practice

29

Why would stochastic gradient ever work???

30

Gradient is direction of steepest ascent

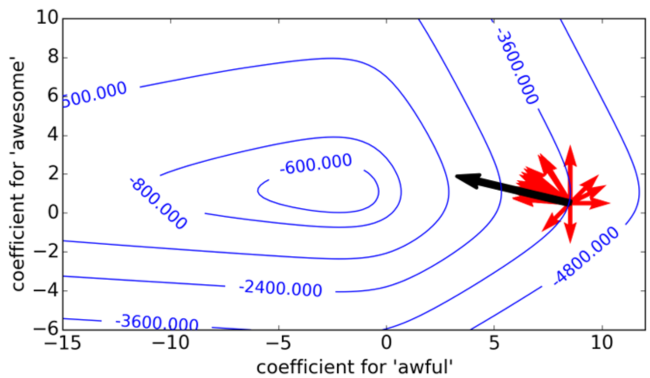


Gradient is "best" direction, but
any direction that goes "up" would be useful

 CS 229: Machine Learning

31

In ML, steepest direction is sum of "little directions" from each data point



Sum over
data points

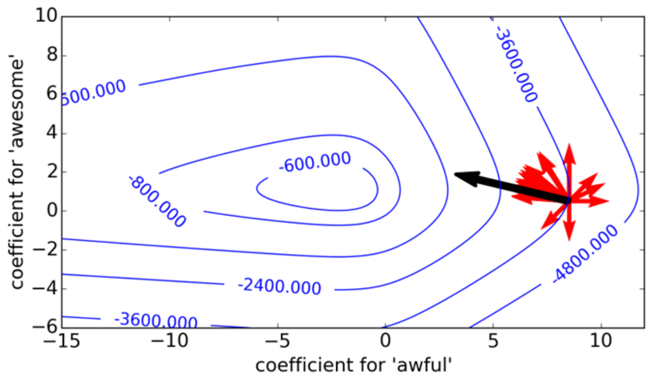
$$\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_j} = \sum_{i=1}^{N} \frac{\partial \ell_i(\mathbf{w})}{\partial \mathbf{w}_j}$$

For most data points,
contribution points "up"

 CS 229: Machine Learning

32

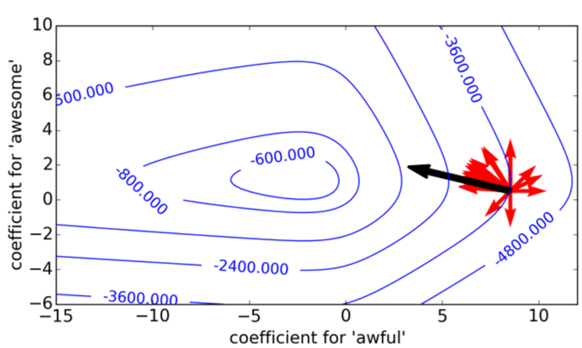
Stochastic gradient:
Pick a data point and move in direction



$$\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_j} \approx \frac{\partial \ell_i(\mathbf{w})}{\partial \mathbf{w}_j}$$

Most of the time, total likelihood will increase

Stochastic gradient ascent:
Most iterations increase likelihood, but sometimes decrease it ➔
On average, make progress



until converged

  for i=1,…,N

    for j=0,…,D

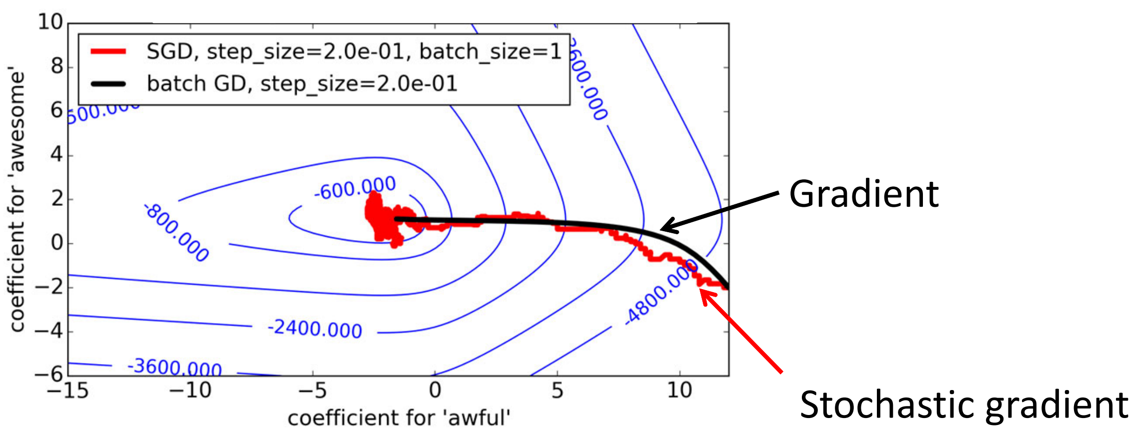
      $w_j^{(t+1)} \leftarrow w_j^{(t)} + \eta \dfrac{\partial \ell_i(\mathbf{w})}{\partial \mathbf{w}_j}$

    $t \leftarrow t + 1$

Convergence path

35

Convergence paths

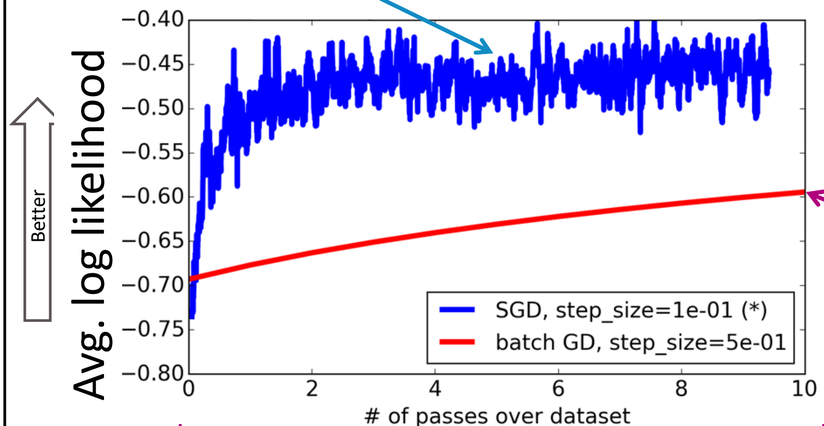


36

Stochastic gradient convergence is "noisy"

Stochastic gradient makes "noisy" progress

Stochastic gradient achieves higher likelihood sooner, but it's noisier



Gradient usually increases likelihood smoothly

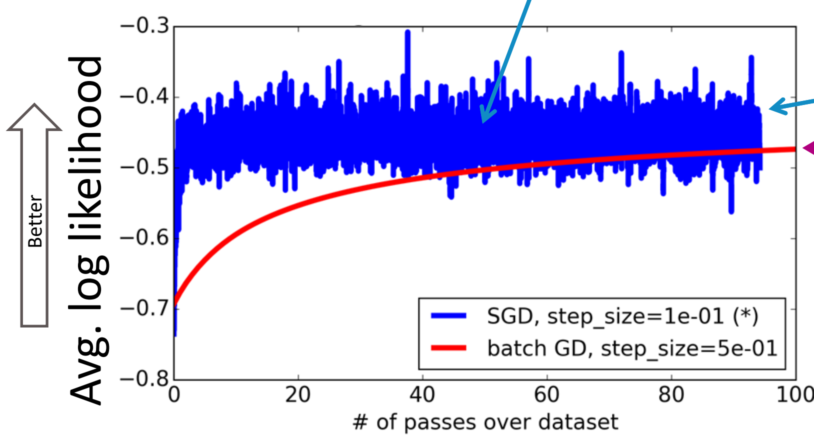
Total time proportional to # passes over data

 CS 229: Machine Learning

37

Eventually, gradient catches up

Note: should only trust "average" quality of stochastic gradient



Stochastic gradient

Gradient

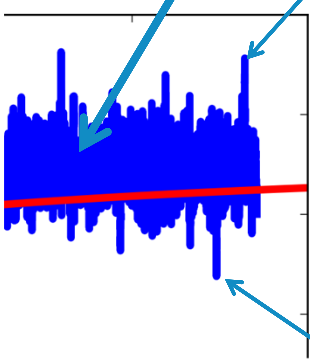
 CS 229: Machine Learning

38

The last coefficients may be really good or really bad!! ☹

Stochastic gradient will eventually oscillate around a solution

$w^{(1005)}$ was good



$w^{(1000)}$ was bad

How do we minimize risk of picking bad coefficients

Minimize noise: don't return last learned coefficients

Output average:

$$\hat{w} = \frac{1}{T} \sum_{t=1}^{T} w^{(t)}$$

©2024 Emily Fox CS 229: Machine Learning

39

Summary of why stochastic gradient works

- Gradient finds direction of steeps ascent
- Gradient is sum of contributions from each data point
- Stochastic gradient uses direction from 1 data point
- On average increases likelihood, sometimes decreases
- Stochastic gradient has "noisy" convergence

©2024 Emily Fox CS 229: Machine Learning

40

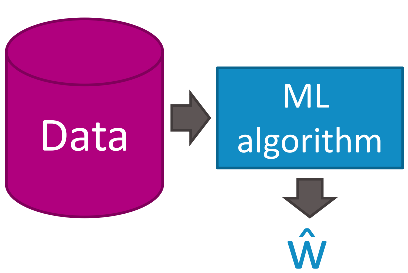
Online learning:
Fitting models from streaming data

41

Batch vs online learning

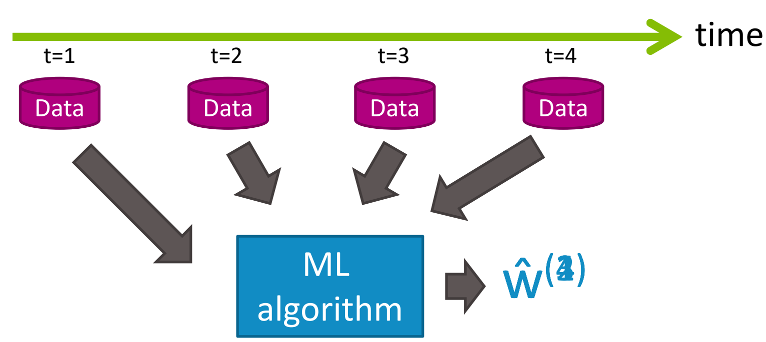
Batch learning

- All data is available at start of training time



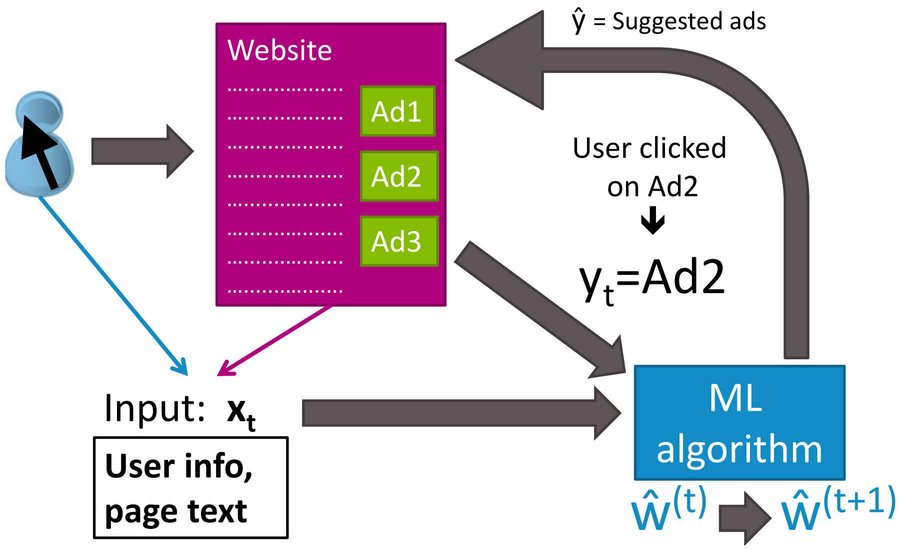
Online learning

- Data arrives (streams in) over time
  - Must train model as data arrives!



42

Online learning example: Ad targeting



©2024 Emily Fox CS 229: Machine Learning

43

Online learning problem

- Data arrives over each time step t:
  - Observe input x_t
    - Info of user, text of webpage
  - Make a prediction $\hat{y}_t$
    - Which ad to show
  - Observe true output y_t
    - Which ad user clicked on

Need ML algorithm to update coefficients each time step!

©2024 Emily Fox CS 229: Machine Learning

44

Stochastic gradient ascent can be used for online learning!!!

- init $w^{(1)}$=0, t=1
- Each time step t:
  - Observe input x_t
  - Make a prediction $\hat{y}_t$
  - Observe true output y_t
  - Update coefficients:

for j=0,…,D

$$w_j^{(t+1)} \leftarrow w_j^{(t)} + \eta \frac{\partial \ell_t(\mathbf{w})}{\partial \mathbf{w}_j}$$

Summary of online learning

Data arrives over time

Must make a prediction every time new data point arrives

Observe true class after prediction made

Want to update parameters immediately

Summary of stochastic gradient descent

47

What you can do now…

- Significantly speedup learning algorithm using stochastic gradient
- Describe intuition behind why stochastic gradient works
- Apply stochastic gradient in practice
- Describe online learning problems
- Relate stochastic gradient to online learning

48

Stochastic gradient descent more formally

OPTIONAL

Learning Problems as Expectations

- Minimizing loss in training data:
  - Given dataset:
    - Sampled iid from some distribution p(**x**) on features:
  - Loss function, e.g., hinge loss, logistic loss,…
  - We often minimize loss in training data:

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

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

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

- 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:
- "True" gradient ascent rule:
- 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:
- What if we estimate gradient with just one sample???
  - Unbiased estimate of gradient
  - Very noisy!
  - Called stochastic gradient ascent (or descent)
    - Among many other names
  - VERY useful in practice!!!

