S&DS 355 / 365 / 565

Data Mining and Machine Learning

Classification and Gradient Descent

Tuesday, September 10

Yale

Office Hours

365:

- Derek Feng: Tuesday 5:00pm-6:00pm, DL #225
- Brandon Chow: Mondays, 6:30pm-7:30pm, 24 Hillhouse (Classroom)
- Colleen Chan: Thursdays 5:00-6:00pm, 17 Hillhouse Rm 7 (Lower Level)
- Kasra Esfandiari: Wednesdays, 6:30pm-7:30pm, 17 Hillhouse Rm 220
- (ULA) Qinying Sun: Friday, 2:00pm-3:00pm, 17 Hillhouse Rm 110

Office Hours

355:

- Parker Holzer: Wednesday 4:00-5:00pm (24 Hillhouse Ave, classroom 107)
- Xinyi Zhong: Thursday 5:00-6:00pm (24 Hillhouse Ave, classroom 107)
- Zifan Li: Monday 7:30 - 8:30pm (24 Hillhouse Ave, classroom 107)

ULAs:

- Max Yuhas: Wednesday 6:00-7:00pm (17 Hillhouse, Room 07)
- Chloe Zhou: Thursday 7:30-8:30pm (Bass L06-A, for September 12)
- Daniel Zhou: Monday TBD
- Adriel Sumathipala: TBD

Outline

- Shrinkage, bias and variance
- Regularization
- Stochastic gradient descent

What did we talk about last time?

- Logistic regression is a linear model of the log-odds
- If data are perfectly separable, log-likelihood will be unbounded

Penalization, shrinkage, bias and variance

Estimator $\widehat{\theta}$ of a parameter θ:

bias^2	$\left(\mathbb{E}(\widehat{\theta}) - \theta\right)^2$
variance	$\mathbb{E}(\widehat{\theta} - \mathbb{E}(\widehat{\theta}))^2$

Penalization, shrinkage, bias and variance

Estimator $\widehat{\theta}$ of a parameter θ:

$$\textit{bias}^2 \qquad \Big(\mathbb{E}(\widehat{\theta}) - \theta\Big)^2$$

$$\textit{variance} \qquad \mathbb{E}(\widehat{\theta} - \mathbb{E}(\widehat{\theta}))^2$$

Expected squared error decomposes as

$$\mathbb{E}(\widehat{\theta} - \theta)^2 = \text{bias}^2 + \text{variance}$$

Penalization, bias and variance

Let's see how shrinkage affects the bias and variance.

Suppose $Y \sim N(\theta, \sigma^2)$.

(a) $\widehat{\theta} = Y$. Bias? Variance?

Penalization, bias and variance

Let's see how shrinkage affects the bias and variance.

Suppose $Y \sim N(\theta, \sigma^2)$.

(a) $\widehat{\theta} = Y$. Bias? Variance?

(b) $\widehat{\theta} = bY$, for $0 \leq b \leq 1$. Bias? Variance?

Exercise

Consider the simplified version of the objective function

$$F(\beta) = (Y - \beta)^2 + \lambda\beta^2$$

What is the minimizer $\widehat{\beta}$?

Exercise

Consider the simplified version of the objective function

$$F(\beta) = (Y - \beta)^2 + \lambda\beta^2$$

What is the minimizer $\widehat{\beta}$?

$$\widehat{\beta} = \left(\frac{1}{1+\lambda}\right) Y$$

Exercise

Now let's add a predictor variable,

$$F(\beta) = (Y - X\beta)^2 + \lambda\beta^2$$

What is the minimizer?

Exercise

Now let's add a predictor variable,

$$F(\beta) = (Y - X\beta)^2 + \lambda\beta^2$$

What is the minimizer?

$$\widehat{\beta} = \frac{XY}{X^2 + \lambda}$$

Shrinkage

In a *shrinkage estimator*, we squash down the estimate by a scaling factor. For example,

$$\widehat{\beta} \leftarrow \left(\frac{1}{1+\lambda}\right)\widehat{\beta}$$

This induces bias-variance tradeoff — the bias goes up, but the variance goes down.

Penalization

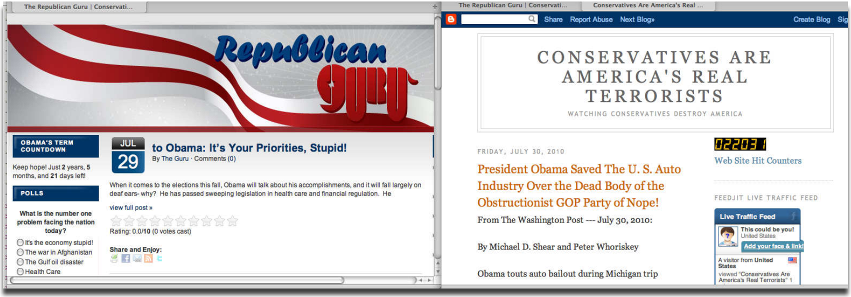
To guard against overfitting, we can *penalize* the coefficients:

$$F(\beta) = -\text{log-likelihood}(\beta) + \lambda\|\beta\|^2$$

- Large coefficients incur a large penalty
- The *regularization parameter* λ controls the tradeoff between fit to the data, and size of the coefficients
- Small λ: high variance, low bias
- Large λ: low variance, high bias
- As λ increases, the size of the coefficients $|\beta_j|$ decreases.

Political blog classification

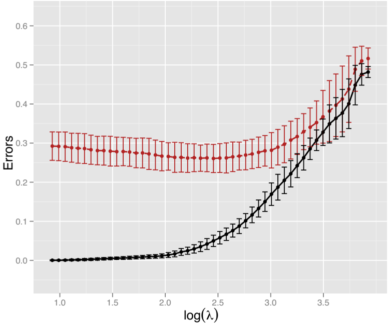
- Political Blog Classification. A collection of 403 political blogs were collected during two months before the 2004 presidential election. The goal is to predict whether a blog is *liberal* ($Y = 0$) or *conservative* ($Y = 1$) given the content of the blog.



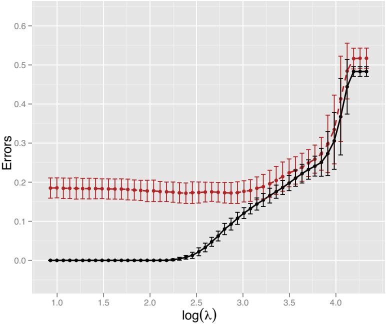
Political blog classification

- 403 blogs
- 205 are “liberal” and 198 are “conservative”
- For each word, value of a feature is word frequency
- Lower case and remove highly frequent words, throw out those appearing fewer than 10 times.
- 23,955 features
- Links to 292 popular blogs included as binary vector

Political blog classification results

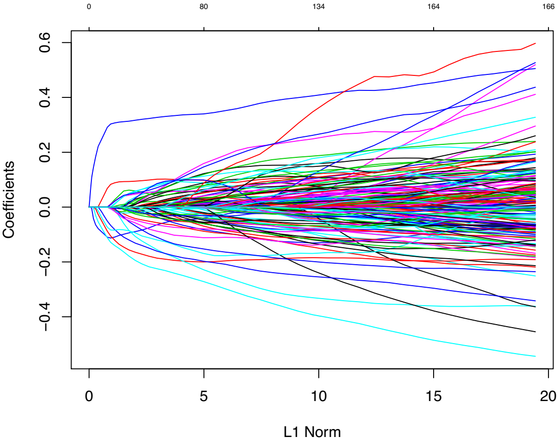


without links



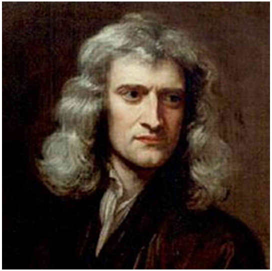
with links

Political blog classification results



regularization paths

Newton's method

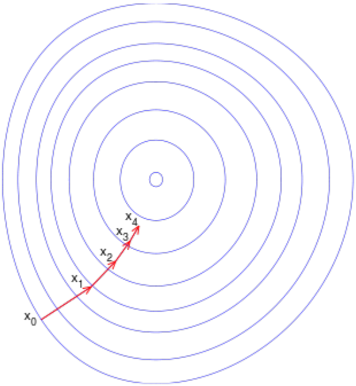


- "Grand-daddy of optimization"
- Fast convergence
- Great properties
- Second order method
- Not scalable

Gradient Descent

Gradient descent is a procedure for finding the arguments that minimize a particular function (called a **cost function**).

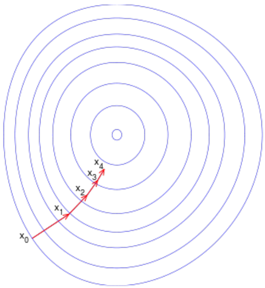
e.g. cost function could be a negative likelihood or negative log-likelihood.



Gradient Descent

Goal: Find $(\theta_1, \ldots, \theta_p)$ that minimizes **cost function** $L(\theta_1, \ldots, \theta_p)$

Update equation:



$$\theta_j \leftarrow \theta_j - \rho \frac{\partial L}{\partial \theta_j}$$

In matrix form:

$$\theta \leftarrow \theta - \rho \nabla L(\theta)$$

ρ is called the learning rate.

don't take too large steps or otherwise risk overshoot

not too small, otherwise too slow

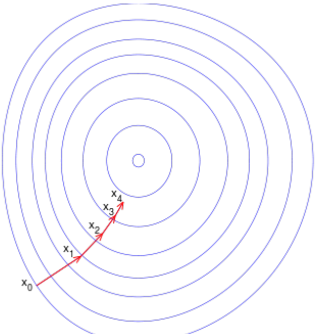
Intuition for GD

355 skip

The gradient gives the direction of *steepest ascent*. Consider the Taylor Series of f about θ,

$$f(\theta + \delta) = f(\theta) + \nabla f(\theta) \cdot \delta + \ldots ,$$

for some small increment δ. Then, ignoring higher order terms, it is clear that to maximize $f(\theta + \delta)$ we must pick δ in the direction of $\nabla f(\theta)$.



GD for Logistic Regression

Cost function (negative log-likelihood):

$$L(\beta) = -\sum \left(y_i x_i^t \beta - \log(1 + e^{x_i^t \beta}) \right)$$

Update for logistic regression:

$$\beta_j \leftarrow \beta_j - \rho \frac{\partial L}{\partial \beta_j}.$$

$$\beta_j \leftarrow \beta_j - \rho \sum_i \left(\frac{e^{x_i^t \beta}}{1 + e^{x_i^t \beta}} - y_i \right) x_{ij}.$$

GD for Linear Regression

Gradient descent also works for linear regression:

$$L(\beta) = \|Y - X\beta\|^2$$

$$\frac{\partial L(\beta)}{\partial \beta_j} = -2\sum_{i=1}^{n}(y_i - x_i^t\beta)x_{ij}$$

GD update step:

$$\beta_j \leftarrow \beta_j + \rho\sum_{i=1}^{n}(y_i - x_i^t\beta)x_{ij}.$$

GD for Linear Regression

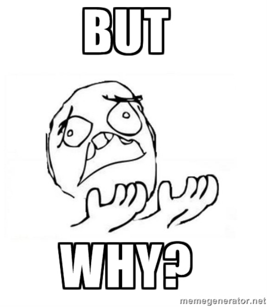
Gradient descent also works for linear regression:

$$L(\beta) = \|Y - X\beta\|^2$$

$$\frac{\partial L(\beta)}{\partial \beta_j} = -2\sum_{i=1}^{n}(y_i - x_i^t\beta)x_{ij}$$

GD update step:

$$\beta_j \leftarrow \beta_j + \rho\sum_{i=1}^{n}(y_i - x_i^t\beta)x_{ij}.$$



Improving Upon Gradient Descent

Each step of (batch) gradient descent requires a calculation involving all of the data points.

Improving Upon Gradient Descent

Each step of (batch) gradient descent requires a calculation involving all of the data points.

Stochastic gradient descent, in contrast, only computes based on a smaller subset of the data points (e.g. 1 observation) at each step.

II: Stochastic gradient descent

- Suppose that we want to fit a really big model, n and d very large
- What is complexity of least squares and IRLS?

II: Stochastic gradient descent

- Suppose that we want to fit a really big model, n and d very large
- What is complexity of least squares and IRLS?

$$O(nd^2 + d^3)$$

- Not practical for large problems
- Second order method, using covariance matrix and Hessian, solving linear system
- How can we get this to scale?

Typical example

- We want to classify documents according to whether or not they are about corporate news.
- There are about 780,000 documents in the collection
- 60,000,000 words
- Document represented as sparse tf-idf vector

$$1 \mid 5 : 1.1789641e{-}01 \quad 39 : 6.0373064e{-}02 \quad 45 : 1.3163488e{-}01$$

- The text takes about 1.1 Gbytes
- How can we efficiently train a classifier?

Online learning

We will introduce a method that

- Reads in the documents one at a time
- Updates the model for each document
- Updates are linear in the size of the document
- Uses little memory, never reads in the entire corpus
- Only stores a dictionary of feature weights

Stochastic gradient descent

We initialize all weights to zero: $\beta_j = 0$, $j = 1, \ldots, d$.

We read through the data one record at a time, and update the model.

1. Read data item x
2. Make a prediction $\widehat{y}(x) = \sum_{j=1}^{d} \beta_j x_j$
3. Observe the true response/label y
4. Update the weights β so $\widehat{y}$ is closer to y

Stochastic gradient descent

To begin, suppose we are just doing linear regression. We initialize all weights to zero: $\beta_j = 0$, $j = 1, \ldots, d$.

We read through the data one record at a time, and update the model.

1. Read data item x
2. Make a prediction $\widehat{y}(x) = \sum_{j=1}^{d} \beta_j x_j$
3. Observe the true response/label y
4. Update the weights β so $\widehat{y}$ is closer to y

$$\beta_j \longleftarrow \beta_j + \eta(y - \widehat{y})x_j$$

Stochastic gradient descent

Think about how to apply this to the data we discussed earlier

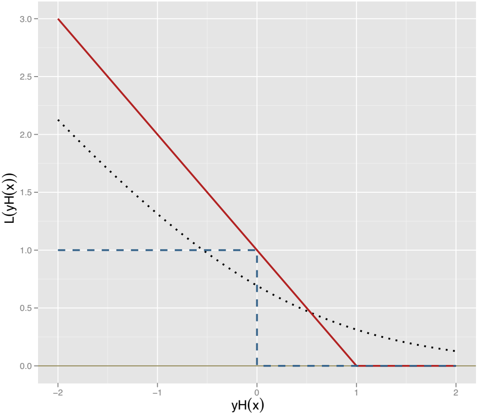
- We want to classify documents according to whether or not they are about corporate news.
- There are about 780,000 documents in the collection
- 60,000,000 words
- Document represented as sparse tf-idf vector

$$1 \mid 5 : 1.1789641e{-}01 \quad 39 : 6.0373064e{-}02 \quad 45 : 1.3163488e{-}01$$

- The text takes about 1.1 Gbytes

Key is that each data item is represented sparsely

SGD for general loss



- Squared error $L(y, \beta^T x) = (y - x^T \beta)^2$
- Logistic: $L(y, \beta^T x) = -yx^T \beta + \log(1 + \exp(x^T \beta))$
- Hinge: $L(y, \beta^T x) = (1 - yx^T \beta)_+$

SGD for general loss

SGD update:

$$\boldsymbol{\beta} \longleftarrow \boldsymbol{\beta} - \eta \nabla L(y, \boldsymbol{\beta}^T x)$$
$$\beta_j \longleftarrow \beta_j - \eta \frac{\partial L(y, \boldsymbol{\beta}^T x)}{\partial \beta_j}$$

- η is the *learning rate* or “step size”
- Needs to be chosen carefully, getting smaller over time

SGD for general loss

SGD update:

$$\boldsymbol{\beta} \longleftarrow \boldsymbol{\beta} - \eta \nabla L(y, \boldsymbol{\beta}^T x)$$
$$\beta_j \longleftarrow \beta_j - \eta \frac{\partial L(y, \boldsymbol{\beta}^T x)}{\partial \beta_j}$$

Some intuition for what this is doing, and why the step size needs to decrease

$$L(\boldsymbol{\beta} + \epsilon v) \approx L(\boldsymbol{\beta}) + \epsilon v^T \nabla L(\boldsymbol{\beta})$$
$$L(\boldsymbol{\beta} - \eta \nabla L(\boldsymbol{\beta})) \approx L(\boldsymbol{\beta}) - \eta \|\nabla L(\boldsymbol{\beta})\|^2$$

This is why SGD is going downhill — if η is small enough

SGD for logistic regression

SGD Update:

$$\beta_j \longleftarrow \beta_j + \eta(y - \pi)x_j$$

$$\beta_j x_j \longleftarrow \beta_j x_j + \eta(y - \pi)x_j^2$$

$$\boldsymbol{\pi} = \frac{1}{1 + \exp(-\boldsymbol{\beta}^T \mathbf{x})}$$

Case checking:

- Suppose $y = 1$ and probability π is high?

SGD for logistic regression

SGD Update:

$$\beta_j \longleftarrow \beta_j + \eta(y - \pi)x_j$$

$$\beta_j x_j \longleftarrow \beta_j x_j + \eta(y - \pi)x_j^2$$

$$\boldsymbol{\pi} = \frac{1}{1 + \exp(-\boldsymbol{\beta}^T \mathbf{x})}$$

Case checking:

- Suppose $y = 1$ and probability π is high? *small change*
- Suppose $y = 1$ and probability π is small?

SGD for logistic regression

SGD Update:

$$\beta_j \longleftarrow \beta_j + \eta(y - \pi)x_j$$

$$\beta_j x_j \longleftarrow \beta_j x_j + \eta(y - \pi)x_j^2$$

$$\boldsymbol{\pi} = \frac{1}{1 + \exp(-\boldsymbol{\beta}^T \mathbf{x})}$$

Case checking:

- Suppose $y = 1$ and probability π is high? *small change*
- Suppose $y = 1$ and probability π is small? *big change* $\uparrow$
- Suppose $y = 0$ and probability π is small?

SGD for logistic regression

SGD Update:

$$\beta_j \longleftarrow \beta_j + \eta(y - \pi)x_j$$

$$\beta_j x_j \longleftarrow \beta_j x_j + \eta(y - \pi)x_j^2$$

$$\boldsymbol{\pi} = \frac{1}{1 + \exp(-\boldsymbol{\beta}^T \mathbf{x})}$$

Case checking:

- Suppose $y = 1$ and probability π is high? *small change*
- Suppose $y = 1$ and probability π is small? *big change* $\uparrow$
- Suppose $y = 0$ and probability π is small? *small change*
- Suppose $y = 0$ and probability π is big?

SGD for logistic regression

SGD Update:

$$\beta_j \longleftarrow \beta_j + \eta(y - \pi)x_j$$

$$\beta_j x_j \longleftarrow \beta_j x_j + \eta(y - \pi)x_j^2$$

$$\boldsymbol{\pi} = \frac{1}{1 + \exp(-\boldsymbol{\beta}^T \mathbf{x})}$$

Case checking:

- Suppose $y = 1$ and probability π is high? *small change*
- Suppose $y = 1$ and probability π is small? *big change* $\uparrow$
- Suppose $y = 0$ and probability π is small? *small change*
- Suppose $y = 0$ and probability π is big? *big change* $\downarrow$

SGD: choice of learning rate

A conservative choice of learning rate is

$$\eta_t = \frac{1}{t}$$

A more agressive choice is

$$\eta_t = \frac{1}{\sqrt{t}}$$

SGD: choice of learning rate

Learning rate should scale as

$$\eta_t = \frac{1}{\sqrt{t}}$$

Problem: Some of the updates may be on different scales.

SGD: choice of learning rate

Learning rate should scale as

$$\eta_t = \frac{1}{\sqrt{t}}$$

Problem: Some of the updates may be on different scales.

Solution: Let $g_{tj} = \dfrac{\partial L(y_t, \beta^T x_t)}{\partial \beta_j}$

Scale gradients to get update rule

$$\beta_j \longleftarrow \beta_j - \eta \frac{g_{tj}}{\sqrt{\sum_{s=1}^{t} g_{sj}^2}}$$

SGD: scaling issues

For a linear model, the SGD update is

$$\beta_j \longleftarrow \beta_j - C_t x_j$$

If x_j increases by a factor of two, the weight β_j should decrease by a factor of two.

This update doesn't respect that scaling

SGD: scaling issues

Usual solution is to "standardize" each variable — subtract out the mean and divide by the standard deviation

$$x_j \leftarrow \frac{x_j - \mathrm{mean}(x_j)}{\sqrt{\mathrm{var}(x_j)}}$$

But this involves "looking ahead" to compute the mean and variance, and destroys the online property of the algorithm

SGD: scaling issues

Usual solution is to "standardize" each variable — subtract out the mean and divide by the standard deviation

$$x_j \leftarrow \frac{x_j - \mathrm{mean}(x_j)}{\sqrt{\mathrm{var}(x_j)}}$$

But this involves "looking ahead" to compute the mean and variance, and destroys the online property of the algorithm

Solution: The mean and variance can be updated in an online manner, in constant time, by storing auxiliary variables for each component j.

SGD: Regularization

A "ridge" penalty $\lambda \sum_{j=1}^{d} \beta_j^2$ is easily handled.

Gradient changes by an additive term $2\lambda\beta_j$. Update becomes

$$\begin{aligned} \beta_j &\longleftarrow \beta_j + \eta\{(y-\pi)x_j - \lambda\beta_j\} \\ &= (1-\eta\lambda)\beta_j + \eta(y-\pi)x_j \end{aligned}$$

$$\beta_j x_j \longleftarrow (1-\eta\lambda)\beta_j x_j + \eta(y-\pi)x_j^2$$

Observe that this "does the right thing" whether β_j wants to be large positive or negative.

- *The penalty shrinks β_j toward zero*

What did we learn today?

- As we penalize $\|\beta\|^2$ from being too big, this *shrinks* the estimated coefficients toward zero.
- Ridge regression shrinks the coefficients. Good for high dimensions.
- If predictor variables are highly correlated, the model estimates may be unstable
- Stochastic gradient descent is a first order method that scales to large classification (and regression) problems
- Choosing the learning rate is a little tricky
- Difficult to parallelize, but not always necessary

Readings

Classification is covered in Chapter 4 of our ISL book. In particular, Section 4.3 is on logistic regression, and Section 4.4 is on linear and quadratic discriminant analysis.