Gradient descent

Adam Richards

Galvanize, Inc

Last updated: 24. April 2017

1. Review
2. Optimization
3. Newton's Method
4. Gradient Descent
5. References

Objectives

- Review
- A bit in linear regression and optimization
- Newton's Method
- Gradient Descent
  - Know when and why to use Gradient Descent
  - Implement the Gradient Descent algorithm and understand different convergence criteria.
  - Know what Stochastic Gradient Descent is and understand its pros and cons.
  - Compare it to Newton's method and understand the pros and cons.

Shrinkage methods

- Motivation - recall that the least squares approach to finding model parameters represents a specific case of maximum likelihood and overfitting is a general property of maximum likelihood estimation (MLE)
- So what again is regularization?

See pages 5-11 in (Bishop, 2006)

Shrinkage methods

- Motivation - recall that the least squares approach to finding model parameters represents a specific case of maximum likelihood and overfitting is a general property of maximum likelihood estimation (MLE)
- So what again is regularization? - technique to control overfitting by introducing an a penalty term to the error function in order to discourage coefficients from reaching large values

$$\widetilde{E}(\mathbf{w}) = \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2 + \lambda \|\mathbf{w}\|^2 \tag{1}$$

where

$$\|\mathbf{w}\|^2 = \boldsymbol{w}^T \boldsymbol{w} = w_0^2 + w_1^2 \ldots w_n^2 \tag{2}$$

Note that λ governs the relative importance of the regularization term compared with the SSE term

See pages 5-11 in (Bishop, 2006)

More on shrinkage methods

- Why do we use the term shrinkage?
- Lasso Regression?
- Ridge Regression?

When we penalize by the sum of square errors in neural networks it is known as weight decay See pages 5-11 in (Bishop, 2006)

More on shrinkage methods

- Why do we use the term shrinkage? Regularization is also referred to as shrinkage because it reduced the values of the coefficients
- Lasso Regression?
- Ridge Regression?

When we penalize by the sum of square errors in neural networks it is known as weight decay See pages 5-11 in (Bishop, 2006)

More on shrinkage methods

- Why do we use the term shrinkage? Regularization is also referred to as shrinkage because it reduced the values of the coefficients
- Lasso Regression?

$$\hat{\mathbf{w}}^{\text{lasso}} = \underset{\mathbf{w}}{\text{argmin}} \left\{ = \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2 + \lambda \|\mathbf{w}\|_1 \right\} \tag{3}$$

where

$$\|w\|_1 = \sum_{j=1}^{M} |w_j|$$

- Ridge Regression?

When we penalize by the sum of square errors in neural networks it is known as weight decay See pages 5-11 in (Bishop, 2006)

More on shrinkage methods

- Why do we use the term shrinkage? Regularization is also referred to as shrinkage because it reduced the values of the coefficients
- Lasso Regression?

$$\hat{\mathbf{w}}^{\text{lasso}} = \underset{\mathbf{w}}{\operatorname{argmin}} \left\{ = \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2 + \lambda \|\mathbf{w}\|_1 \right\} \tag{3}$$

where

$$\|w\|_1 = \sum_{j=1}^{M} |w_j|$$

- Ridge Regression?

$$\hat{\mathbf{w}}^{\text{ridge}} = \underset{\mathbf{w}}{\operatorname{argmin}} \left\{ = \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2 + \lambda \|\mathbf{w}\|_2^2 \right\} \tag{4}$$

$$\|w\|_2 = \sum_{j=1}^{M} w_j^2$$

When we penalize by the sum of square errors in neural networks it is known as weight decay See pages 5-11 in (Bishop, 2006)

L1 and L2 penalties

Interpretation

When two predictors are highly correlated L1 penalties tend to pick one of the two while L2 will take both and shrink the coefficients

- In general L1 penalties are better at recovering sparse signals
- L2 penalties are better at minimizing prediction error

- So what type of regression is good for eliminating correlated variables?
- And if I just want to reduce the influence of two correlated variables?
- But what I just do not know which to use?

Logistic regression

Some perspective

Fisher proposed linear discriminant analysis in 1936. In the 1940s, various authors put forth an alternative approach, logistic regression. In the early 1970s, Nelder and Wedderburn coined the term generalized linear models for an entire class of statistical learning methods that include both linear and logistic regression as special cases. (Hastie et al., 2009) pp20.

Why might linear regression not be appropriate for the following?

- `y_label={1:'asthma',2:'lung cancer',3:'bronchitis'}`
- In logistic regression we are trying to model the probabilities of the K classes via linear functions in x
- These models are usually fit by MLE
- Rather than model the response directly (like in linear regression) logistic regression models the probability that Y belongs to a category
- e.g $P(\text{asthma} \mid \text{years\_smoked})$ is between 0 and 1 for any `years_smoked`

Objectives

✓ Review

- A bit in linear regression and optimization
- Newton's Method
- Gradient Descent
  - Know when and why to use Gradient Descent
  - Implement the Gradient Descent algorithm and understand different convergence criteria.
  - Know what Stochastic Gradient Descent is and understand its pros and cons.
  - Compare it to Newton's method and understand the pros and cons.

Optimization methods

Objective function

Any function for which we wish to find the minimum or maximum

In logistic regression the log-likelihood (prob. parameters given the data) for N observations can be specified as

$$\ell(\beta) = \sum_{i=1}^{N} \{y_i \log p(x_i;\beta) + (1 - y_i) \log(1 - p(x_i;\beta))\} \tag{5}$$

where $p(x;\beta)$ and $1 - p(x;\beta)$ are the probabilities of class 1 and class 2 in a $k = 2$ class scenario.
Recall that we wish to model $p(X)$ using the logistic function

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}} \text{ or } \frac{p(X)}{1 - p(X)} = e^{\beta_0+\beta_1 X} \tag{6}$$

If $p(X) = 0.2$ then $1/5$ people will have asthma with an odds of $\frac{0.2}{1-0.2} = \frac{1}{4}$.

(James et al., 2014) Chapter 4

Take the log of both sides of our logistic function then we get the logit or log-odds

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X \tag{7}$$

- How do we interpret β_1 in a linear regression setting?
- How do we interpret β_1 in a logistic regression setting?

We want to find $\hat{\beta}_0$ and $\hat{\beta}_1$ s.t. plugging in estimates for

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}} \tag{8}$$

close to 1 for individuals with asthma and close to 0 for those without

(James et al., 2014) Chapter 4

Take the log of both sides of our logistic function then we get the logit or log-odds

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X \tag{7}$$

- How do we interpret β_1 in a linear regression setting?
  β_1 gives the average change in Y associated with a one-unit increase in X
- How do we interpret β_1 in a logistic regression setting?

We want to find $\hat{\beta}_0$ and $\hat{\beta}_1$ s.t. plugging in estimates for

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}} \tag{8}$$

close to 1 for individuals with asthma and close to 0 for those without

(James et al., 2014) Chapter 4

Take the log of both sides of our logistic function then we get the logit or log-odds

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X \tag{7}$$

- How do we interpret β_1 in a linear regression setting?
  β_1 gives the average change in Y associated with a one-unit increase in X
- How do we interpret β_1 in a logistic regression setting?
  Increasing X by one unit changes the log odds by β_1

We want to find $\hat{\beta}_0$ and $\hat{\beta}_1$ s.t. plugging in estimates for

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}} \tag{8}$$

close to 1 for individuals with asthma and close to 0 for those without

(James et al., 2014) Chapter 4

A step back

What are we trying to accomplish in optimization?

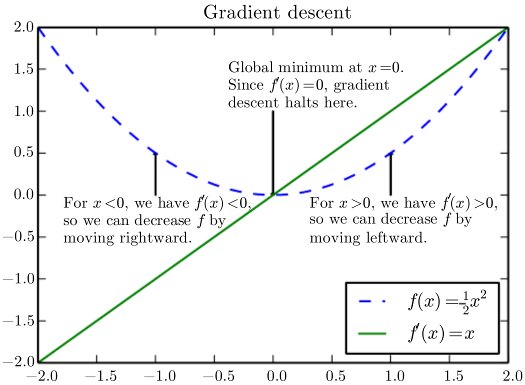
1. Find the parameters of a model which maximize the likelihood of data
2. Find the parameters of a model which minimize a cost function

Cost function

- Maybe there is a cost (quality of care) associated with the total number of patients in an emergency room
- We are interested in predicting profit at a business and there is some cost associated with producing the product

Optimization refers to the task of either minimizing or maximizing some function $f(\mathbf{x})$ by altering $\mathbf{x}$. This function is called an objective function and if we are minimizing it has several names: cost function, loss function, error function.

Gradient-based optimization



The derivatives of a function can be used to follow a function to its minimum. (Bengio et al., 2016)[Figure 4.1]

Calculus notation

	Calculus
$\frac{dy}{dx}$	Derivative of y with respect to x
$\frac{\partial y}{\partial x}$	Partial derivative of y with respect to x
$\nabla_{\boldsymbol{x}} y$	Gradient of y with respect to $\boldsymbol{x}$
$\nabla_{\boldsymbol{X}} y$	Matrix derivatives of y with respect to $\boldsymbol{X}$
$\nabla_{\mathsf{X}} y$	Tensor containing derivatives of y with respect to X
$\frac{\partial f}{\partial \boldsymbol{x}}$	Jacobian matrix $\boldsymbol{J} \in \mathbb{R}^{m \times n}$ of $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$
$\nabla^2_{\boldsymbol{x}} f(\boldsymbol{x})$ or $\boldsymbol{H}(f)(\boldsymbol{x})$	The Hessian matrix of f at input point $\boldsymbol{x}$
$\int f(\boldsymbol{x}) d\boldsymbol{x}$	Definite integral over the entire domain of $\boldsymbol{x}$
$\int_{\mathbb{S}} f(\boldsymbol{x}) d\boldsymbol{x}$	Definite integral with respect to $\boldsymbol{x}$ over the set $\mathbb{S}$

Sympy can by useful for checking your calculus

A simple derivative

```
import sympy
x = sympy.symbols('x')
sympy.diff(4*x**2)
```

```
8*x
```

A gradient

```
m=sympy.Matrix(sympy.symbols('x y'))
[sympy.diff(sum(m*m.T), i) for i in m]
```

```
 [2*x + 2*y, 2*x + 2*y]
```

Why?

Suppose we have a function $y = f(x)$, where both x and y are real numbers. The *derivative* of this function is denoted as $f'(x)$ or as $\frac{dy}{dx}$. The derivative $f'(x)$ gives the slope of $f(x)$ at the point x. In other words, it specifies how to scale a small change in the input in order to obtain the corresponding change in the output: $f(x + \epsilon) \approx f(x) + \epsilon f'(x)$.

(Bengio et al., 2016)

Objectives

- ✓ Review
- ✓ A bit in linear regression and optimization
- Newton's Method
- Gradient Descent
  - Know when and why to use Gradient Descent
  - Implement the Gradient Descent algorithm and understand different convergence criteria.
  - Know what Stochastic Gradient Descent is and understand its pros and cons.
  - Compare it to Newton's method and understand the pros and cons.

Optimization

To maximize the log-likelihood

$$\ell(\beta) = \sum_{i=1}^{N} \{y_i \log p(x_i;\beta) + (1-y_i)\log(1-p(x_i;\beta))\} \tag{9}$$

We set the derivatives to 0

$$\frac{\partial \ell(\beta)}{\partial \beta} = \sum_{i=1}^{N} x_i(y_i - p(x_i;\beta)) = 0 \tag{10}$$

These are known as score equations and to solve these we can use the Newton-Raphson algorithm, which makes use of a second-derivative or Hessian matrix.

$$\frac{\partial^2 \ell(\beta)}{\partial\beta\partial\beta^T} = -\sum_{i=1}^{N} x_i x_i^T p(x_i;\beta)(1-p(x_i;\beta)) \tag{11}$$

(Hastie et al., 2009) Chapter 4

Newton update

$$\beta^{\text{new}} = \left(\beta^{\text{old}} - \frac{\partial^2 \ell(\beta)}{\partial\beta\partial\beta^T}\right)^{-1} \frac{\partial \ell(\beta)}{\partial\beta} \tag{12}$$

It is convenient to write this as matrix notation. Let $\mathbf{y}$ be the vector of y_i, $\mathbf{X}$ is a $N \times (p+1)$ matrix of x values, $\mathbf{p}$ is a vector of fitted probabilities and $\mathbf{W}$ is a N x N diagonal matrix of weights with the ith element being $p(x_i|\beta^{\text{old}})(1 - p(x_i|\beta^{\text{old}}))$.

$$\frac{\partial \ell(\beta)}{\partial\beta} = \mathbf{X}^T(\mathbf{y} - \mathbf{p}) \tag{4.24}$$

$$\frac{\partial^2 \ell(\beta)}{\partial\beta\partial\beta^T} = -\mathbf{X}^T\mathbf{W}\mathbf{X} \tag{4.25}$$

The Newton step is thus

$$\begin{aligned} \beta^{\text{new}} &= \beta^{\text{old}} + (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{y} - \mathbf{p}) \\ &= (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}\left(\mathbf{X}\beta^{\text{old}} + \mathbf{W}^{-1}(\mathbf{y} - \mathbf{p})\right) \\ &= (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}\mathbf{z}. \end{aligned} \tag{4.26}$$

In the second and third line we have re-expressed the Newton step as a weighted least squares step, with the response

$$\mathbf{z} = \mathbf{X}\beta^{\text{old}} + \mathbf{W}^{-1}(\mathbf{y} - \mathbf{p}), \tag{4.27}$$

This is known as iteratively reweighed least squares (Hastie et al., 2009) Chapter 4

Objectives

- ✓ Review
- ✓ A bit in linear regression and optimization
- ✓ Newton's Method
- Gradient Descent
  - Know when and why to use Gradient Descent
  - Implement the Gradient Descent algorithm and understand different convergence criteria.
  - Know what Stochastic Gradient Descent is and understand its pros and cons.
  - Compare it to Newton's method and understand the pros and cons.

Gradient descent

There are three main variants of gradient descent

1. Batch gradient descent - computes the gradient of the cost function w.r.t θ for the entire data set
2. Stochastic gradient descent (SGD) - performs gradient descent for each training example in x along with its corresponding y.
3. Mini-batch gradient descent - performs an update for every mini-batch of training examples

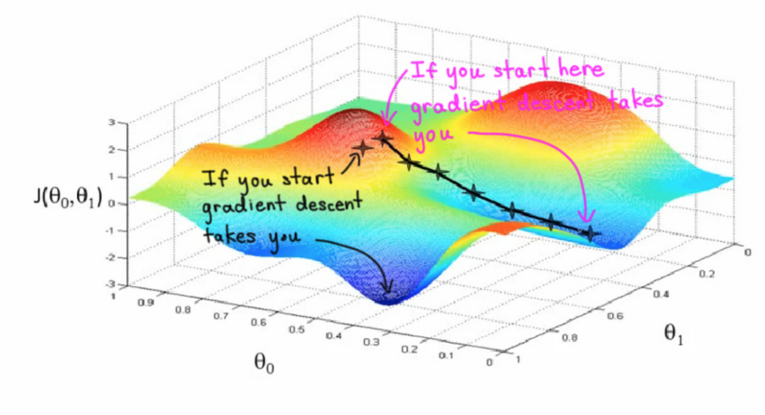
http://sebastianruder.com/optimizing-gradient-descent/

Using intuition

- The gradient is the multivariate analogue of the derivative.
- Geometrically the gradient is the direction of steepest descent.

$$\nabla f = \sum_{i=1}^{P} \frac{\partial f}{\partial x_i} e_i \tag{13}$$

- How can we use the direction of steepest descent to find the minimum of our function?
- How about the maximum?


If you start here
gradient descent takes
you
If you start
gradient descent
takes you
$J(\theta_0,\theta_1)$
θ_0
θ_1
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1

Gradient Descent Algorithm

$$\theta_{i+1,j} = \theta_{i,j} + \alpha \frac{\partial J((\vec{\theta}_i))}{\partial \theta_{i,j}}$$

Parameters

- α: Learning rate
- i: Iteration
- j: Feature

Convergence

- Max number of iterations
- Magnitude of gradient
- Change in cost function

$$\frac{(cost_{old} - cost_{new})}{cost_{old}} < \epsilon$$

Pseudocode

```
new_params = dict((i,0) for i in range(k))
while not has_converged:
    params = copy(new_params)

    for theta in params:
        update = alpha * gradient(theta, params)
        new_params[theta] -= update
```

Things to remember

- Needs a differentiable cost/likelihood function
- Local maxima/minima
- Need to scale the features
- Although SGD seems nicer it can have performance issues associated with oscillation

Why does it matter to me?

- Under the hood
- Neural Networks
- Economics
- ...

```python
import sklearn.linear_model as lm
help(lm.LogisticRegression)
```

Gradient descent vs Newton's Method

Which to use?

	Gradient descent	Newton's Method
Simplicity	a bit less	a bit more
Parameters	α	None
Iterations	more	less
n $<$ 1000	same	same
n $>$ 10000	better	worse

This is taken from a video done by Andrew Ng—have a look for more details
https://www.youtube.com/watch?v=iwO0JPt59YQ

Objectives

- ✓ Review
- ✓ Optimization
- ✓ Newton's Method
- ✓ Gradient Descent
  - ✓ Know when and why to use Gradient Descent
  - ✓ Implement the Gradient Descent algorithm and understand different convergence criteria.
  - ✓ Know what Stochastic Gradient Descent is and understand its pros and cons.
  - ✓ Compare it to Newton's method is and understand the pros and cons.

Where do I go from here?

- Checkout the starter code and read carefully the assignment
- Andrew Ng lecture - https://www.youtube.com/watch?v=5u4G23_OohI
- For more rigor you can read section pp.238-243 in (Bishop, 2006)
- My calculus is terrible can't Python help me? SymPy calculus
- A blog post that uses sympy to discuss MLE bases optimization in Bernoulli trials

References I

Bengio, Y., Goodfellow, I. J., and Courville, A. (2016). *Deep Learning*. MIT Press. Book in preparation for MIT Press.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2014). *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated.