Machine Learning – Gradient Descent

Faculty:
Farshid Alizadeh-Shabdiz, PhD, MBA
Sept 2021

Gradient Descent

- Most important and main approach to train
  - machine learning algorithms, in general
  - And neural networks, specifically

What is Covered Here

- Linear Regression – a simple base to introduce the Gradient Descent
- Gradient Descent
- Different variations of Gradient Descent

Regression

- Regression Function

$$f(x) = E(y \mid x = x_i)$$

- Regression Function minimizes mean squared error (MSE)

Regression – How to calculate *f*

- The conditional probability cannot be calculated
- Relax the definition and let calculate the conditional probability for a small region
- Nearest Neighbor or local averaging (which also provide smooth solution)

$$f(x) = E(y \mid x \in [x_i - \Delta, x + \Delta])$$

Regression

- Linear model of

$$f(x) = a_0 + a_1 X_1 + a_2 X_2 + \ldots + a_m X_m$$

- Polynomial regression, for example a quadratic model will look like below

$$f(x) = a_0 + a_1 X + a_2 X^2$$

$$f(x) = a_0 + a_1 X_1 + a_2 X_1^2 + b_1 X_2 + b_2 X_2^2$$

Linear Regression

- Linear model

$$f(x) = a_0 + a_1 X_1 + a_2 X_2 + \ldots + a_m X_m$$

- Very simple algorithm – maybe the simplest!

Linear Regression – Closed form

- Linear model

$$y = f(x) = a_0 + a_1 X_1 + a_2 X_2 + \dots + a_m X_m$$

Closed form answer – Normal Equation

$$A = (X^T X)^{-1} X^T Y$$

- Fitness function: Mean Squared Error (MSE
- Computational complexity $O(n^{2.4})$ to $O(n^3)$
  - $O(n^{2.4}) = 5.3$ and $O(n^3) = 8$
- There are numerical solutions to find matrix inverse efficiently - e.g. PseudoInverse.

Error Surface

- Squared error

$$E(x) = \frac{1}{N}\sum_{i=1}^{N}(y_i - (a_0 + a_1 X_i))^2$$

- Error surface of a linear system with squared error is a quadratic bowl

Gradient Descent

- Gradient descent can be applied to linear and non-linear systems
- High level steps of Gradient Descent
  - Initialize parameters value
  - Change value of parameters in the direction of gradient
  - Move toward local minimum value

- The gradient is calculated as partial derivative of *f(x)* respect to a_0 and a_1.

$$a_i(t) = a_i(t-1) - \varepsilon_i \frac{\partial E(x)}{\partial a_i}$$

- ε_i Called learning rate or step size

SKYHOOK®

Gradient Descent Initial Value

- Movement in the direction of gradient –
  - Small issue – slow on elongated elliptical surfaces
- Initial value is a randomly selected number Most common one is staring from the origin (0,0,…0)
- Gradient descent will also settle at the local minimum
- Different initial value might result to different minimum
- Therefore, not a unique answer

SKYHOOK®

Gradient Descent Issues

- Historically – gradient descent was used to train each layer separately by using layer-wise greedy training. Took a long time and not stable!

- Local minima
  - Despite complex error surface, local minima is not an issue
  - Many people tried to show and prove why
- Flat surface
  - Min and max
  - Saddle points
- Gradient into the wrong direction

Gradient Descent Notes

- Simultaneous update or not?
  - Gradient descent correct approach is updating all the parameters simultaneously
- Data normalization
  - Feature scaling
  - Feature shifting
  - De-correlation
- Second order methods have been researched, but nothing in the practical realm yet! Keep your eyes open for that.

Gradient Descent

- Batch Gradient Descent
- Stochastic Gradient Descent
- Mini-batch Gradient Descent

Stochastic Gradient Descent

- The most popular learning method
- Unbiased estimate with not a large variance
- Hyper-parameters
  - Step size
  - Random selection method (with or without replacement). Almost all without replacement is used
- More sensitive to step size
- Behavior
  - Area of confusion
    - Help with not overfitting!
- Early stopping a good option
- Simulation from internet
  - https://towardsdatascience.com/why-gradient-descent-isnt-enough-a-comprehensive-introduction-to-optimization-algorithms-in-59670fd5c096

Stochastic Gradient Descent(SGD)

- SGD will help
  - Getting out of local minima
  - Might not settle in real minimum
    - Simulated Annealing: reduce step size as we get closer to minimum

- Learning schedule is a big topic here
  - Too quick reduction of step size, results to stock in local minimum
  - Too slow reduction of step size, results to jumpiness around the minimum

Epoch

- Epoch definition:
  - Each round of *M* iteration is called epoch
  - Or one iteration of running the entire training is called an Epoch

Stochastic Gradient Descent

- Random sampling has to be representative of the general population. This is called *Stratified* Sampling, in which population is grouped to homogeneous set called Strata, and samples are selected from each Strata according to general population.
  - For example the US population is 48.7% female and 51.3% male

Stochastic Gradient Descent – Momentum-Based

- Momentum added or momentum-based

$$a_i(t+1) = a_i(t) - \Lambda(t)$$

$$\Lambda(t) = \gamma\Lambda(t-1) + \varepsilon_i \frac{\partial E(x_t)}{\partial a_i(t)}$$

- Which accelerates in the direction of consistent gradient
- Start with small gamma (e.g. 0.5), since gradient might be large. As gradient goes down, increase gamma toward its final value (e.g. 0.9)
- Maybe order of magnitude faster than gradient descent!
- Issue: Oscillates in the valley

Stochastic Gradient Descent – Nesterov Accelerated Gradient

- Resolves Momentum GD problem
- Gradient of the destination point is used to correct the gradient

$$a_{temp} = a(t) - \gamma\Lambda(t-1)$$

$$a(t+1) = a_{temp} - \varepsilon \frac{\partial E}{\partial a_{temp}}$$

$$\Lambda(t) = \gamma\Lambda(t-1) + \varepsilon \frac{\partial E}{\partial a_{temp}}$$

- NAG is much faster than Momentum-based.
- Less oscillation compared to momentum-based.

Learning Rate Adaptation Version of Gradient Descent

Stochastic Gradient Descent – ADAGrad

- Addressing the issue of moving fast in the direction of steepest descent, which might not be optimum

$$\mathrm{S(t)} = S(t-1) + \frac{\partial E}{\partial a(t)}\frac{\partial E}{\partial a(t)}$$

$$a_{temp} = a(t) - \varepsilon \frac{\partial E}{\partial a(t)}\frac{1}{\sqrt{S(t)+\epsilon}}$$

The above equation is calculated for each component of a

- General equation (in which $\otimes$ is element wise multiplication)

$$\vec{S}(\mathrm{t}) = \vec{S}(t-1) + \frac{\partial E}{\partial a(t)} \otimes \frac{\partial E}{\partial a(t)}$$

$$a_{temp} = a(t) - \varepsilon \frac{\partial E}{\partial a(t)} \otimes \frac{1}{\sqrt{S(t)+\epsilon}}$$

SKYHOOK®

RMSProp

- Scale the learning rate by running average of recent gradients
- Because of using MSE, it is more sensitive to large gradients

$$\mathrm{S(t)} = \gamma S(t-1) + (1-\gamma)\frac{\partial E}{\partial a(t)}\frac{\partial E}{\partial a(t)}$$

$$a_{temp} = a(t) - \varepsilon \frac{\partial E}{\partial a(t)} \frac{1}{\sqrt{S(t)+\epsilon}}$$

AdaGrad slows down too fast!
RMSProp addresses that issue.

Adam – Adaptive Moment Estimation

- RMSProp + Momentum Optimization
  - Like momentum: exponential decaying avg of past gradients
  - Like RMSProp: gradients scale with decaying average of past squared gradients

  Note that *t* is number of iterations

$$\Lambda(t+1) = \beta_1 \Lambda(t) - (1-\beta_1)\frac{\partial E}{\partial a(t)}$$

$$\vec{S}(\mathrm{t}) = \beta_2 \vec{S}(t-1) + (1-\beta_2)\frac{\partial E}{\partial a(t)} \otimes \frac{\partial E}{\partial a(t)}$$

$$\Lambda(t+1) = \Lambda(t)\,/(1\text{-}\beta_1{}^t)$$

$$\overrightarrow{S'} = \vec{S}/(1-\beta_2{}^t)$$

$$a(t+1) = a(t) + \varepsilon\Lambda(t+1) \otimes \frac{1}{\sqrt{\overrightarrow{S'}(t) + \epsilon}}$$

Adam – Adaptive Moment Estimation – cont'

Note about Adam

- Equation 3 and 4 – a minor change to the algorithm. Since *MSE* and *S* are small at the beginning, these equations help to boost their value at the beginning
- Typical value of $\boldsymbol{\beta}_1$ is 0.9 and $\boldsymbol{\beta}_2$ is 0.99
- Typical value of $\boldsymbol{\varepsilon}$ is 10^{-10} to prevent divide by zero

AdaMax

- A variation of Adam, but it is more stable (depending on the data set)

$$\Lambda(t+1) = \beta_1 \Lambda(t) - (1-\beta_1)\frac{\partial E}{\partial a(t)}$$

$$\vec{S}(\mathrm{t}) = \max\{\beta_2 \vec{S}(t-1), \frac{\partial E}{\partial a(t)}\}$$

$$\Lambda(t+1) = \Lambda(t) / (1 - \beta_1{}^t)$$

$$a(t+1) = a(t) + \varepsilon\Lambda(t+1) \otimes \frac{1}{\overrightarrow{S'}(t) + \epsilon}$$

- Note:

$$\vec{S}(\mathrm{t}) = \beta_2 \vec{S}(t-1) + (1-\beta_2)\frac{\partial E}{\partial a(t)} \otimes \frac{\partial E}{\partial a(t)}$$

Nadam

- It is Adam with Nestrov – as a result it converges faster than Adam.

Adaptive Learning Rate

- Adjust the rate based on consistency of the gradient.
  - If gradient remains consistent, increase the learning rate, and if they are not consistent, reduce the learning rate
- To avoid noisy gradient, big mini-batch sizes

$$\Delta a_i(t) = -\varepsilon_i g_i \frac{\partial E}{\partial a_i(t)}$$

$$Initial\_value\_g_i = 1$$

$$if \ \frac{\partial E}{\partial a_i(t)} \times \frac{\partial E}{\partial a_i(t-1)} > 0,$$

$$g_i(t) = g_i(t-1) + 0.05$$

$$Else: g_i(t) = g_i(t-1) \times 0.95$$

Adaptive Learning Rate + Momentum

- (Jacobs 1989) suggested combining adaptive learning rate with momentum gradient by using agreement between current gradient and accumulated gradient

Regularization

Learning Curves

- Polynomial Regression can easily over-fit the data
- Learning curve (RMSE vs training set size) can help us to detect over-fitting and under-fitting
  - If training data and validation data are far apart, that is an indication of over-fitting
  - If training data and validation data are performing poorly and they are the same, that is an indication of under-fitting

Regularization

- Ridge Regression

$$E(x) = MSE(a) + \frac{\alpha}{2}\sum_{i=1}^{N}(a_i)^2$$

- Lasso Regression

$$E(x) = MSE(a) + \alpha\sum_{i=1}^{N}|a_i|$$

- Elastic Regression

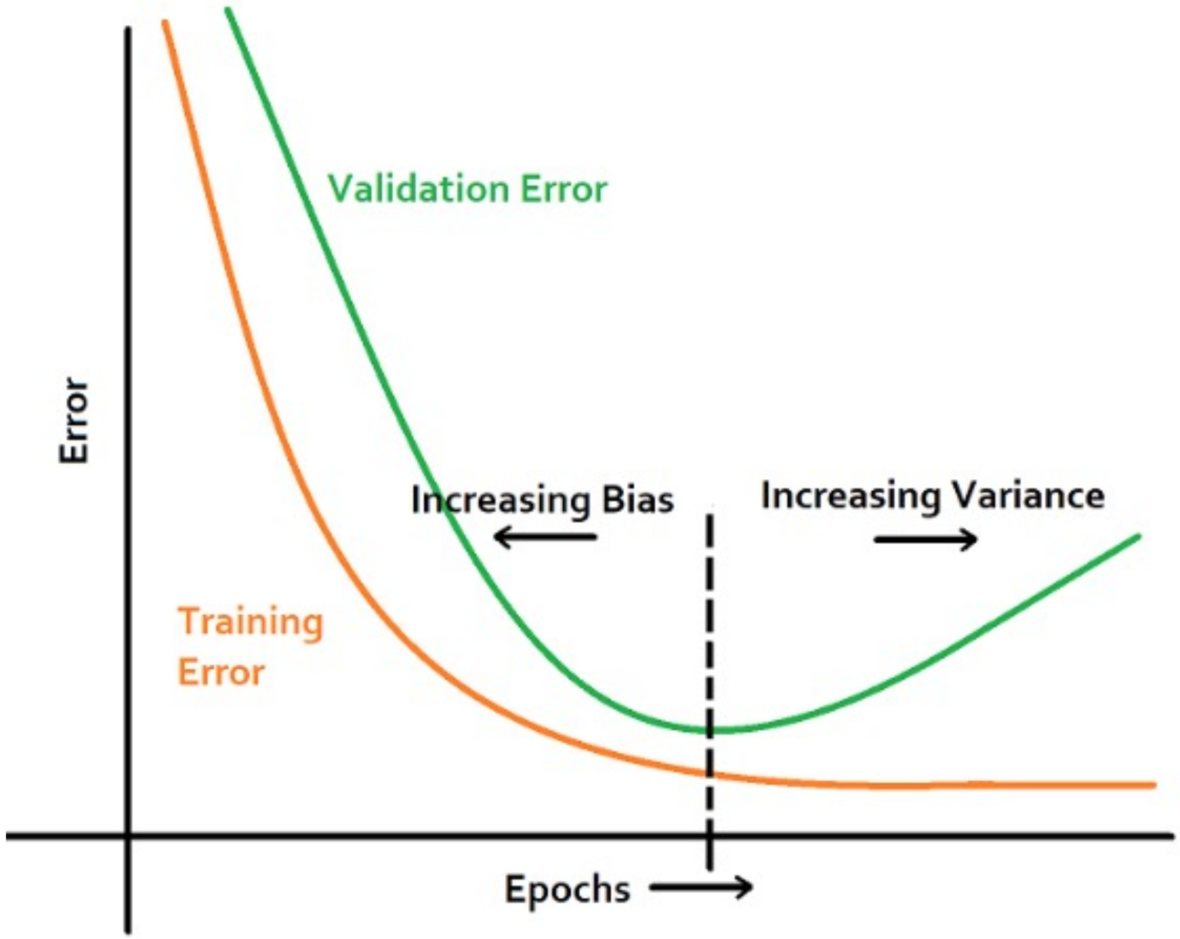
$$E(x) = MSE(a) + r\alpha\sum_{i=1}^{N}|a_i| + \frac{(1-r)\alpha}{2}\sum_{i=1}^{N}(a_i)^2$$

SKYHOOK

Regularization Sensitivity

- Regularization is sensitive to data scale
  - Therefor normalization is important

Regularization – Early Stopping



- For simple linear model with a quadratic error function, early stopping and Ridge regularization are equivalent