Linear Models
Gradient descent (intro)

Srinivasan Viswanathan

Nov 2, 2022

Recap

- Recap of previous lecture.
- Intro to machine learning of linear models
- Curse of dimensionality
- Intro to gradient descent.
- Assignment discussion for few min at the end.

Linear Models

- $Y=w0+w1x+w2x^2+w3x^4$ …
- Parameter fit by minimizing the objective function RMS.
- Example – live demo.
- Observations
  - -

Curse of Dimensionality

- Distribution of data points as dimension increases ?
- Volume of cube ?
- Fraction of sphere ?
- Demo.
- How many parameters required as dimensions increase ?
- Observations-

Gradient Descent

- Convex Function
- RMS objective functions are convex.

- F''(x) Is > 0
- F(ax1+(1-a)x2) <= aF(x1)+(1-a)F(x2)
- Algorithm is simple:
  - Move towards the minimum point
  - X(n+1) = x(n)-(learning_rate)*(gradient(Xn))
  - Partial derivatives..
  - Remember these derivatives are wrt parameters not the variable X or y

Gradient Descent

- Initialize
- Keep iterating until max iterations
- Or no changes in parameteers .
- Learning rate has an implication on the workings of the gradient descent.

Linear Models and Gradient Descent

- The function you are trying to arrive at is called as a Hypothesis Function. (Family of functions with different parameters).
- The form of the function for linear model is given below.

$$h_\theta(x^{(i)}) = \theta_0 x_0^{(i)} + \theta_1 x_1^{(i)} + \ldots + \theta_j x_j^{(i)}$$

- The x above could be replaced with $\phi(x)$ more generally.

Linear Models and Gradient Descent (ctd)

- The Cost Function is given below:.

$$J_\theta = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

- The gradient is calculated for every parameter θ_j

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

Linear Models and Gradient Descent (ctd)

- Algorithm:
  Initialization of the Model parameters followed by iteration to convergence.
- At each iteration the parameters are updated with the following formula:

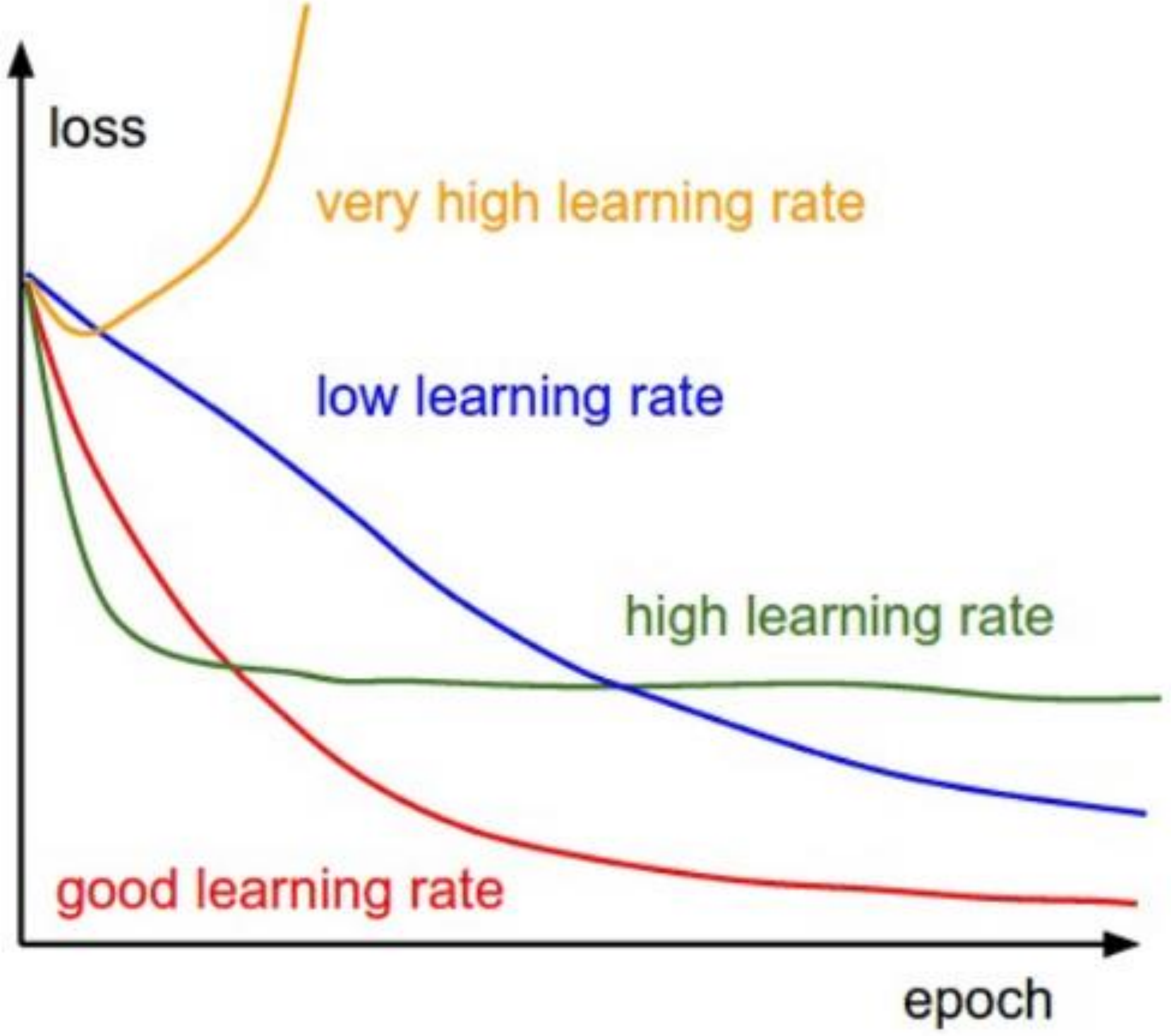
$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Linear Models and Gradient Descent (ctd)

- Convergence Criteria:
  - Max iterations reached
  - Difference in objective minimization (cost) is too small to compute more.
- Gradient_descent()
  - W=initial values (parameters) W=w0, w1, … wn-1
  - While iterations < max OR costdiff > epsilon:
    - W=W-learning_rate*(gradient)
    - Calculate cost again

Linear Models and Gradient Descent (ctd)

- Convergence Criteria:
  - Max iterations reached
  - Difference in objective minimization (cost) is too small to compute more.
- Gradient_descent()
  - W=initial values (parameters) W=w0, w1, … wn-1
  - While iterations < max OR costdiff > epsilon:
    - W=W-learning_rate*(gradient)
    - Calculate cost again


loss
very high learning rate
low learning rate
high learning rate
good learning rate
epoch

Stochastic Gradient Descent

- We pick a random point and do the calculation of the gradient descent only for that point. And recalculate parameters W.

  W = W – learning_rate(gradient of only that random point)
- Results in quick convergence but some times results in more noise and variance.
- Computationally faster, as we compute gradient only for one random point.
- An improvement over Stochastic gradient descent is to use Mini-batch gradient descent for a subset of points, reducces computation burden but improves accuracy over stochastic case.

Linear Models and Gradient Descent (ctd)

- Add Regularizers to objective function to minimize over-fitting, and reduce the values of the coefficients. (Called as Ridge Regression)

$$J_\theta = \frac{1}{2m}\sum_{i=1}^{n}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2} \parallel w \parallel^2$$

  lambda – Regularizer parameter

- Along with learning rate, lambda should also be adjusted.
-

Gradient descent Used in:

- Support vector machines for classification (next class)
- Used in neural networks models.

Must think through the following

- No. Of parameters for a 3 feature vectors and polynomial order is 2 ?
  - Can you generalize the above ?
- Maximum likelihood/MAP(Maximum Posterior) assuming the linear model with noise as gauissian (as we did in the lab) will lead to the same equations as RMS error function plus regularizer ! Investigate.
- Give the formula for the gradient using numpy now:
- Give the formula for the cost function using numpy now :
- Find the difference in performance doing the above two without using numpy ! (similar to our matrix assignment prior).

Must think through the following (ctd.)

- Can we do the linear model for sin(2*pi*x) using gradient desccent ? You will win favorite student tag if you do this.

$$h_\theta(x^{(i)}) = \theta_0 x_0^{(i)} + \theta_1 x_1^{(i)} + \ldots + \theta_j x_j^{(i)}$$

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Thank You !

Srinivasan Viswanathan