(Sub)gradient Descent

CMSC 422

MARINE CARPUAT
marine@cs.umd.edu

Figures credit: Piyush Rai

Amy Webb @amywebb

Follow

The salt lines for tonight's storm is confusing the Tesla's autopilot

RETWEETS 444 LIKES 680

7:17 AM - 13 Mar 2017

38 444 680

Recap: Linear Models

- General framework for binary classification
- Cast learning as optimization problem
- Optimization objective combines 2 terms
  - loss function: measures how well classifier fits training data
  - Regularizer: measures how simple classifier is
- Does not assume data is linearly separable
- Lets us separate model definition from training algorithm **(Gradient Descent)**

Casting Linear Classification as an Optimization Problem

Objective function

Loss function measures how well classifier fits training data

Regularizer prefers solutions that generalize well

$$\min_{\mathbf{w},b} L(\mathbf{w}, b) = \min_{\mathbf{w},b} \sum_{n=1}^{N} \mathbb{I}(y_n(\mathbf{w}^T\mathbf{x}_n + b) < 0) + \lambda R(\mathbf{w}, b)$$

$\mathbb{I}(.)$ Indicator function: 1 if (.) is true, 0 otherwise

The loss function above is called the 0-1 loss

Gradient descent

- A general solution for our optimization problem

$$\min_{\mathbf{w},b} L(\mathbf{w},b) = \min_{\mathbf{w},b} \sum_{n=1}^{N} \mathbb{I}(y_n(\mathbf{w}^T\mathbf{x}_n + b) < 0) + \lambda R(\mathbf{w},b)$$

- Idea: take iterative steps to update parameters in the direction of the gradient

Gradient descent algorithm

Objective function to minimize

Number of steps

Step size

Algorithm 22 $\textsc{GradientDescent}(\mathcal{F}, K, \eta_1, \ldots)$

$z^{(0)} \leftarrow \langle 0, 0, \ldots, 0 \rangle$ // initialize variable we are optimizing

for $k = 1 \ldots K$ **do**

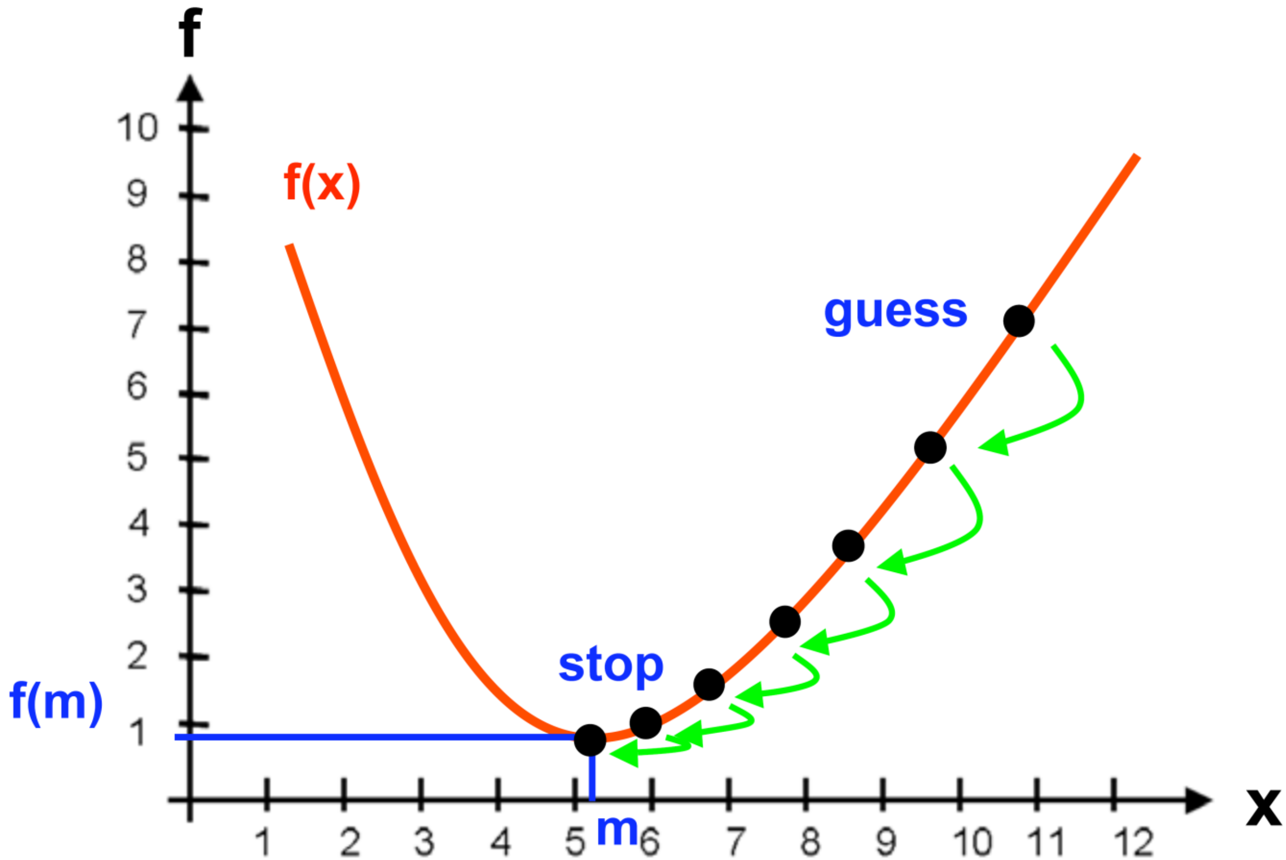
  $g^{(k)} \leftarrow \nabla_z \mathcal{F}|_{z^{(k-1)}}$ // compute gradient at current location

  $z^{(k)} \leftarrow z^{(k-1)} - \eta^{(k)} g^{(k)}$ // take a step down the gradient

end for

return $z^{(K)}$

Illustrating gradient descent in 1-dimensional case



Gradient Descent

- 2 questions
  - When to stop?
    - When the gradient gets close to zero
    - When the objective stops changing much
    - When the parameters stop changing much
    - Early
    - When performance on held-out dev set plateaus
  - How to choose the step size?
    - Start with large steps, then take smaller steps

Now let's calculate gradients for multivariate objectives

- Consider the following learning objective

$$\mathcal{L}(\boldsymbol{w}, b) = \sum_n \exp\left[-y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b)\right] + \frac{\lambda}{2} ||\boldsymbol{w}||^2$$

- What do we need to do to run gradient descent?

(1) Derivative with respect to b

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial b} &= \frac{\partial}{\partial b} \sum_n \exp\left[- y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) \right] + \frac{\partial}{\partial b} \frac{\lambda}{2} \left|\left|\boldsymbol{w}\right|\right|^2 && (6.12) \\
&= \sum_n \frac{\partial}{\partial b} \exp\left[- y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) \right] + 0 && (6.13) \\
&= \sum_n \left(\frac{\partial}{\partial b} - y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) \right) \exp\left[- y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) \right] && (6.14) \\
&= - \sum_n y_n \exp\left[- y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) \right] && (6.15)
\end{aligned}
$$

(2) Gradient with respect to w

$$\nabla_w \mathcal{L} = \nabla_w \sum_n \exp\left[-y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b)\right] + \nabla_w \frac{\lambda}{2} ||\boldsymbol{w}||^2 \tag{6.16}$$

$$= \sum_n \left(\nabla_w - y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b)\right) \exp\left[-y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b)\right] + \lambda \boldsymbol{w} \tag{6.17}$$

$$= -\sum_n y_n \boldsymbol{x}_n \exp\left[-y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b)\right] + \lambda \boldsymbol{w} \tag{6.18}$$

Subgradients

- Problem: some objective functions are not differentiable everywhere
  - Hinge loss, l1 norm

- Solution: subgradient optimization
  - Let's ignore the problem, and just try to apply gradient descent anyway!!
  - we will just differentiate by parts…

Example: subgradient of hinge loss

For a given example n

$$\boldsymbol{\partial}_w \max\{0, 1 - y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b)\} \tag{6.22}$$

$$= \boldsymbol{\partial}_w \begin{cases} 0 & \text{if } y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) > 1 \\ y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) & \text{otherwise} \end{cases} \tag{6.23}$$

$$= \begin{cases} \boldsymbol{\partial}_w 0 & \text{if } y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) > 1 \\ \boldsymbol{\partial}_w y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) & \text{otherwise} \end{cases} \tag{6.24}$$

$$= \begin{cases} \boldsymbol{0} & \text{if } y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) > 1 \\ y_n \boldsymbol{x}_n & \text{otherwise} \end{cases} \tag{6.25}$$

Subgradient Descent for Hinge Loss

Algorithm 23 HingeRegularizedGD($\mathbf{D}$, λ, *MaxIter*)

$\boldsymbol{w} \leftarrow \langle 0, 0, \dots 0 \rangle \quad , \quad b \leftarrow 0$ // initialize weights and bias
for $iter = 1 \dots MaxIter$ **do**
    $\boldsymbol{g} \leftarrow \langle 0, 0, \dots 0 \rangle \quad , \quad g \leftarrow 0$ // initialize gradient of weights and bias
    for all $(\boldsymbol{x}, y) \in \mathbf{D}$ **do**
        if $y(\boldsymbol{w} \cdot \boldsymbol{x} + b) \leq 1$ **then**
            $\boldsymbol{g} \leftarrow \boldsymbol{g} + y\, \boldsymbol{x}$ // update weight gradient
            $g \leftarrow g + y$ // update bias derivative
        end if
    end for
    $\boldsymbol{g} \leftarrow \boldsymbol{g} - \lambda \boldsymbol{w}$ // add in regularization term
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \boldsymbol{g}$ // update weights
    $b \leftarrow b + \eta g$ // update bias
end for
return $\boldsymbol{w}$, b

What is the perceptron optimizing?

Algorithm 5 PerceptronTrain($\mathbf{D}$, *MaxIter*)

$w_d \leftarrow 0$, for all $d = 1 \ldots D$ // initialize weights
$b \leftarrow 0$ // initialize bias
for $iter = 1 \ldots MaxIter$ **do**
  for all $(x,y) \in \mathbf{D}$ **do**
    $a \leftarrow \sum_{d=1}^{D} w_d \, x_d + b$ // compute activation for this example
    if $ya \leq 0$ **then**
      $w_d \leftarrow w_d + y x_d$, for all $d = 1 \ldots D$ // update weights
      $b \leftarrow b + y$ // update bias
    end if
  end for
end for
return $w_0, w_1, \ldots, w_D, b$

- Loss function is a variant of the hinge loss

$$\max\{0, -y_n(\mathbf{w}^T \mathbf{x}_n + b)\}$$

Recap: Linear Models

- Lets us separate model definition from training algorithm **(Gradient Descent)**

Summary

- Gradient descent
  - A generic algorithm to minimize objective functions
  - Works well as long as functions are well behaved (ie convex)
  - Subgradient descent can be used at points where derivative is not defined
  - Choice of step size is important

- Optional: can we do better?
  - For some objectives, we can find closed form solutions (see CIML 6.6)