Slides adapted from Prof Carpuat and Duraiswami

CMSC 422 Introduction to Machine Learning

Lecture 14 (Sub)gradient Descent

Furong Huang / furongh@cs.umd.edu

UNIVERSITY OF MARYLAND

Regrading deadline extended

March 29 11:00 am

Submit along with you exam sheets

Don’t forget to put your name

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 $\textsc{GradientDescent}(\mathcal{F}, K, \eta_1, \dots)$

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

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

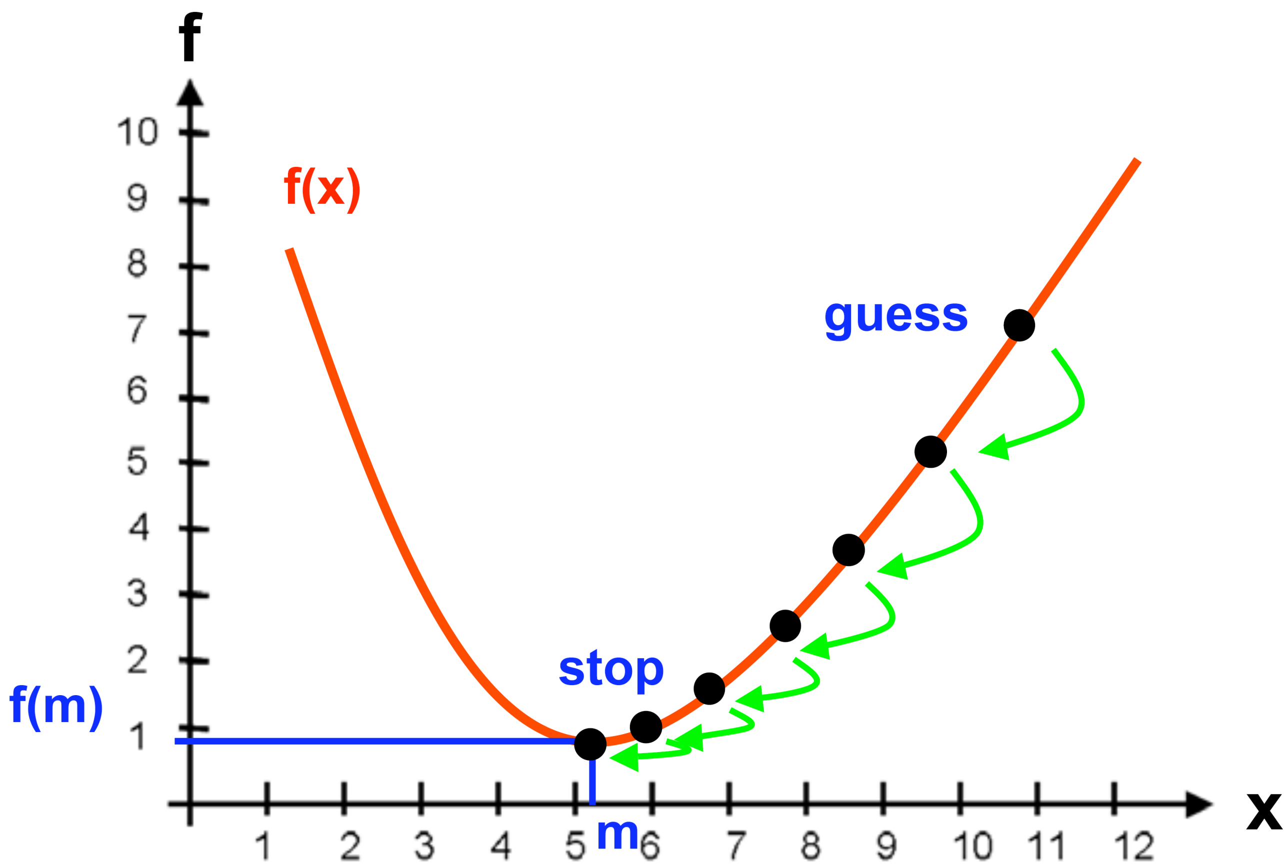
  $g^{(k)} \leftarrow \nabla_z \mathcal{F}|_{z^{(k-1)}}$ // compute gradient at current location

  $z^{(k)} \leftarrow z^{(k-1)} - \eta^{(k)} g^{(k)}$ // 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

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial}{\partial b} \sum_n \exp\left[-y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b)\right] + \frac{\partial}{\partial b} \frac{\lambda}{2} \left|\left|\boldsymbol{w}\right|\right|^2 \tag{6.12}$$

$$= \sum_n \frac{\partial}{\partial b} \exp\left[-y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b)\right] + 0 \tag{6.13}$$

$$= \sum_n \left(\frac{\partial}{\partial b} - y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b)\right) \exp\left[-y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b)\right] \tag{6.14}$$

$$= -\sum_n y_n \exp\left[-y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b)\right] \tag{6.15}$$

(2) Gradient with respect to w

$$\nabla_{w}\mathcal{L} = \nabla_{w}\sum_{n}\exp\big[-y_n(\boldsymbol{w}\cdot\boldsymbol{x}_n + b)\big] + \nabla_{w}\frac{\lambda}{2}\,||\boldsymbol{w}||^2 \tag{6.16}$$

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

$$= -\sum_{n}y_n\boldsymbol{x}_n\exp\big[-y_n(\boldsymbol{w}\cdot\boldsymbol{x}_n + b)\big] + \lambda\boldsymbol{w} \tag{6.18}$$

Subgradients

Problem: some objective functions are not differentiable everywhere

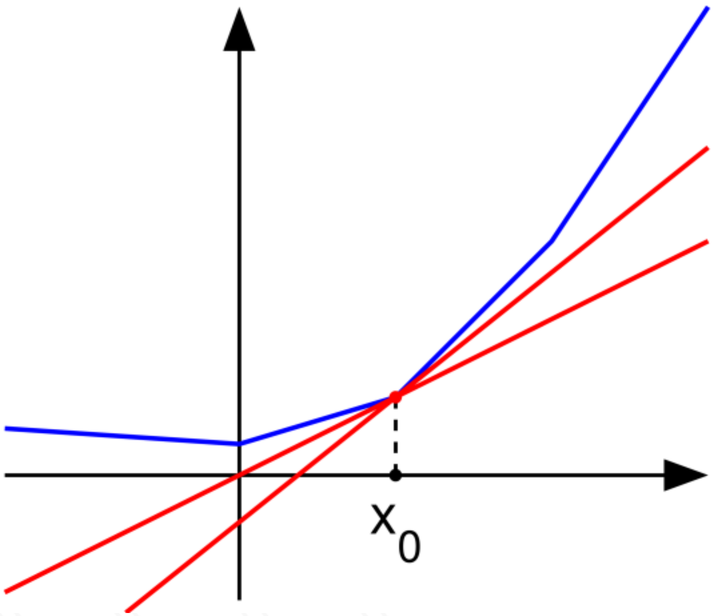
- Hinge loss, l1 norm

Solution: subgradient optimization

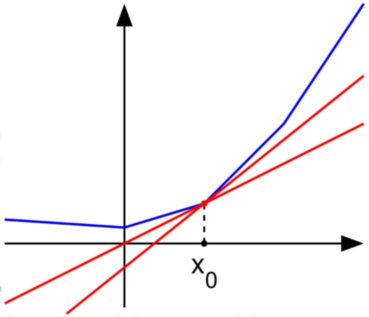
- Let's ignore the problem, and just try to apply gradient descent anyway!!
- we will just differentiate by parts…

Subgradient Review

- Subgradient generalized the derivative to functions which are not differentiable.
- For any x_0 in the domain of the function one can draw a line which goes through the point $(x_0, f(x_0))$ and which is everywhere either **touching** or **below** the graph of f.
- Set-valued



Subgradient Review



Rigorously, a *subgradient* of a convex function $f: I \to \mathbf{R}$ at a point x_0 in the open interval I is a real number c such that

$$f(x) - f(x_0) \geq c(x - x_0)$$

for all x in I. One may show that the set of subgradients at x0 for a convex function is a nonempty closed interval [a, b], where a and b are the one-sided limits.

$$a = \lim_{x \to x_0^-} \frac{f(x) - f(x_0)}{x - x_0}$$

$$b = \lim_{x \to x_0^+} \frac{f(x) - f(x_0)}{x - x_0}$$

which are guaranteed to exist and satisfy a ≤ b.

- The set [a, b] of all subgradients is called the subgradients of the function f at x_0. Since f is convex, if its subdifferential at x_0 contains exactly one subgradient, then f is differentiable at x_0.

Approximating the 0-1 loss with surrogate loss functions

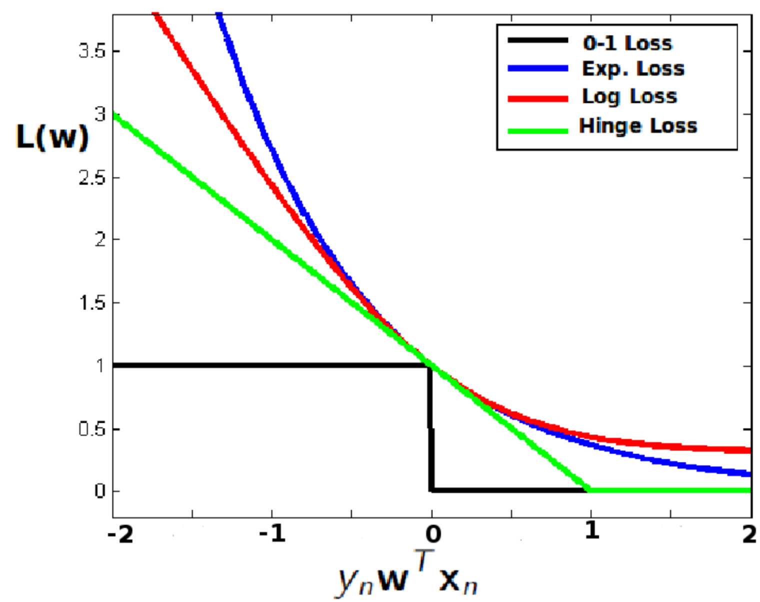
Examples (with b = 0)

Hinge loss $[1 - y_n \mathbf{w}^T \mathbf{x}_n]_+ = \max\{0, 1 - y_n \mathbf{w}^T \mathbf{x}_n\}$

Log loss $\log[1 + \exp(-y_n \mathbf{w}^T \mathbf{x}_n)]$

Exponential loss $\exp(-y_n \mathbf{w}^T \mathbf{x}_n)$

What if $b \neq 0$?



Example: subgradient of hinge loss

For a given example n

$$
\begin{aligned}
&\boldsymbol{\partial}_{w} \max\{0, 1 - y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b)\} && (6.22)\\
&= \boldsymbol{\partial}_{w} \begin{cases} 0 & \text{if } y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) > 1 \\ y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) & \text{otherwise} \end{cases} && (6.23)\\
&= \begin{cases} \boldsymbol{\partial}_{w} 0 & \text{if } y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) > 1 \\ \boldsymbol{\partial}_{w} y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) & \text{otherwise} \end{cases} && (6.24)\\
&= \begin{cases} \boldsymbol{0} & \text{if } y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) > 1 \\ y_n \boldsymbol{x}_n & \text{otherwise} \end{cases} && (6.25)
\end{aligned}
$$

Subgradient Descent for Hinge Loss

Algorithm 23 $\textsc{HingeRegularizedGD}(\mathbf{D}, \lambda, \mathit{MaxIter})$

$\boldsymbol{w} \leftarrow \langle 0, 0, \ldots 0 \rangle \quad , \quad b \leftarrow 0$ // initialize weights and bias

for $\mathit{iter} = 1 \ldots \mathit{MaxIter}$ **do**

  $\boldsymbol{g} \leftarrow \langle 0, 0, \ldots 0 \rangle \quad , \quad g \leftarrow 0$ // 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 $\textsc{PerceptronTrain}(\mathbf{D}, \mathit{MaxIter})$

$w_d \leftarrow 0$, for all $d = 1 \dots D$ // initialize weights
$b \leftarrow 0$ // initialize bias
for $iter = 1 \dots MaxIter$ **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**
      $w_d \leftarrow w_d + yx_d$, for all $d = 1 \dots D$ // update weights
      $b \leftarrow b + y$ // update bias
    end if
  end for
end for
return $w_0, w_1, \dots, w_D, b$

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)

UNIVERSITY OF
MARYLAND

Furong Huang
3251 A.V. Williams, College Park, MD 20740
301.405.8010 / furongh@cs.umd.edu