MONASH University

ETC3555

Statistical Machine Learning

(Stochastic) gradient descent

14 August 2018

Outline

How to minimize E_{in}

How to minimize E_{in}

For logistic regression,

$$E_{\text{in}}(\mathbf{w}) \;=\; \frac{1}{N}\sum_{n=1}^{N} \ln\left(1 + e^{-y_n \mathbf{w}^{\mathsf{T}}\mathbf{x}_n}\right) \qquad \longleftarrow \text{iterative solution}$$

Compare to linear regression:

$$E_{\text{in}}(\mathbf{w}) \;=\; \frac{1}{N}\sum_{n=1}^{N} (\mathbf{w}^{\mathsf{T}}\mathbf{x}_n - y_n)^2 \qquad \longleftarrow \text{closed-form solution}$$

Gradient descent

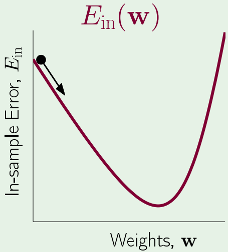
Iterative method: gradient descent

General method for nonlinear optimization

Start at $\mathbf{w}(0)$; take a step along steepest slope

Fixed step size: $\mathbf{w}(1) = \mathbf{w}(0) + \eta\, \hat{\mathbf{v}}$

What is the direction $\hat{\mathbf{v}}$?



$\hat{\boldsymbol{v}}$ is a unit vector, i.e. $\|\hat{\boldsymbol{v}}\| = 1$ and $\eta > 0$

What is the best direction $\hat{\boldsymbol{v}}$?

We want to minimize

$$E_{\text{in}}(\boldsymbol{w}(1)) - E_{\text{in}}(\boldsymbol{w}(0)) = E_{\text{in}}(\boldsymbol{w}(0) + \eta\hat{\boldsymbol{v}}) - E_{\text{in}}(\boldsymbol{w}(0)).$$

The Taylor expansion of $E_{\text{in}}(\boldsymbol{w})$ at $\boldsymbol{w} = \boldsymbol{w}(0) + \eta\hat{\boldsymbol{v}}$ is given by

$$E_{\text{in}}(\boldsymbol{w}(0) + \eta\hat{\boldsymbol{v}}) = E_{\text{in}}(\boldsymbol{w}(0)) + \eta\nabla E_{\text{in}}(\boldsymbol{w}(0))^T\hat{\boldsymbol{v}} + O(\eta^2),$$

where $\nabla E_{\text{in}}(\boldsymbol{w}) = (\frac{\partial E_{\text{in}}(\boldsymbol{w})}{\partial w_1}, \frac{\partial E_{\text{in}}(\boldsymbol{w})}{\partial w_2}, \ldots, \frac{\partial E_{\text{in}}(\boldsymbol{w})}{\partial w_d})^T$ is the gradient.

We can write

$$\begin{aligned} & E_{\text{in}}(\boldsymbol{w}(0) + \eta\hat{\boldsymbol{v}}) - E_{\text{in}}(\boldsymbol{w}(0)) \\ &= \underbrace{E_{\text{in}}(\boldsymbol{w}(0)) + \eta\nabla E_{\text{in}}(\boldsymbol{w}(0))^T\hat{\boldsymbol{v}} + O(\eta^2)}_{\text{Taylor approximation}} - E_{\text{in}}(\boldsymbol{w}(0)) \\ &\approx \eta\nabla E_{\text{in}}(\boldsymbol{w}(0))^T\hat{\boldsymbol{v}}. \end{aligned}$$

What is the best direction $\hat{v}$?

We want to minimize

$$E_{\text{in}}(\boldsymbol{w}(1)) - E_{\text{in}}(\boldsymbol{w}(0)) = E_{\text{in}}(\boldsymbol{w}(0) + \eta \hat{\boldsymbol{v}}) - E_{\text{in}}(\boldsymbol{w}(0)).$$

The Taylor expansion of $E_{\text{in}}(\boldsymbol{w})$ at $\boldsymbol{w} = \boldsymbol{w}(0) + \eta \hat{\boldsymbol{v}}$ is given by

$$E_{\text{in}}(\boldsymbol{w}(0) + \eta \hat{\boldsymbol{v}}) = E_{\text{in}}(\boldsymbol{w}(0)) + \eta \nabla E_{\text{in}}(\boldsymbol{w}(0))^T \hat{\boldsymbol{v}} + O(\eta^2),$$

where $\nabla E_{\text{in}}(\boldsymbol{w}) = (\frac{\partial E_{\text{in}}(\boldsymbol{w})}{\partial w_1}, \frac{\partial E_{\text{in}}(\boldsymbol{w})}{\partial w_2}, \ldots, \frac{\partial E_{\text{in}}(\boldsymbol{w})}{\partial w_d})^T$ is the gradient.

We can write

$$\begin{aligned} & E_{\text{in}}(\boldsymbol{w}(0) + \eta \hat{\boldsymbol{v}}) - E_{\text{in}}(\boldsymbol{w}(0)) \\ &= \underbrace{E_{\text{in}}(\boldsymbol{w}(0)) + \eta \nabla E_{\text{in}}(\boldsymbol{w}(0))^T \hat{\boldsymbol{v}} + O(\eta^2)}_{\text{Taylor approximation}} - E_{\text{in}}(\boldsymbol{w}(0)) \\ &\approx \eta \nabla E_{\text{in}}(\boldsymbol{w}(0))^T \hat{\boldsymbol{v}}. \end{aligned}$$

What is the best direction $\hat{v}$?

We want to minimize

$$E_{\text{in}}(\mathbf{w}(1)) - E_{\text{in}}(\mathbf{w}(0)) = E_{\text{in}}(\mathbf{w}(0) + \eta\hat{\mathbf{v}}) - E_{\text{in}}(\mathbf{w}(0)).$$

The Taylor expansion of $E_{\text{in}}(\mathbf{w})$ at $\mathbf{w} = \mathbf{w}(0) + \eta\hat{\mathbf{v}}$ is given by

$$E_{\text{in}}(\mathbf{w}(0) + \eta\hat{\mathbf{v}}) = E_{\text{in}}(\mathbf{w}(0)) + \eta\nabla E_{\text{in}}(\mathbf{w}(0))^T\hat{\mathbf{v}} + O(\eta^2),$$

where $\nabla E_{\text{in}}(\mathbf{w}) = \left(\frac{\partial E_{\text{in}}(\mathbf{w})}{\partial w_1}, \frac{\partial E_{\text{in}}(\mathbf{w})}{\partial w_2}, \ldots, \frac{\partial E_{\text{in}}(\mathbf{w})}{\partial w_d}\right)^T$ is the gradient.

We can write

$$\begin{aligned} & E_{\text{in}}(\mathbf{w}(0) + \eta\hat{\mathbf{v}}) - E_{\text{in}}(\mathbf{w}(0)) \\ &= \underbrace{E_{\text{in}}(\mathbf{w}(0)) + \eta\nabla E_{\text{in}}(\mathbf{w}(0))^T\hat{\mathbf{v}} + O(\eta^2)}_{\text{Taylor approximation}} - E_{\text{in}}(\mathbf{w}(0)) \\ &\approx \eta\nabla E_{\text{in}}(\mathbf{w}(0))^T\hat{\mathbf{v}}. \end{aligned}$$

What is the best direction $\hat{\boldsymbol{v}}$?

$$\underset{\hat{\boldsymbol{v}}, \|\hat{\boldsymbol{v}}\|=1}{\text{minimize}} \;\; \eta \nabla E_{\text{in}}(\boldsymbol{w}(0))^T \hat{\boldsymbol{v}} = \|\nabla E_{\text{in}}(\boldsymbol{w}(0))\| \, \|\hat{\boldsymbol{v}}\| \cos(\theta)$$

where θ is the angle between $\nabla E_{\text{in}}(\boldsymbol{w}(0))$ and $\hat{\boldsymbol{v}}$.

$$\equiv \underset{\hat{\boldsymbol{v}}}{\text{minimize}} \;\; \|\nabla E_{\text{in}}(\boldsymbol{w}(0))\| \cos(\theta) \equiv \underset{\hat{\boldsymbol{v}}}{\text{minimize}} \;\; \cos(\theta)$$

This quantity is minimized with $\theta = 180^\circ$ (cos(θ) = -1), i.e. $\hat{\boldsymbol{v}}$ is pointing in the opposite direction of the gradient. Since $\hat{\boldsymbol{v}}$ is a unit vector, we have $\hat{\boldsymbol{v}} = -\frac{\nabla E_{\text{in}}(\boldsymbol{w}(0))}{\|\nabla E_{\text{in}}(\boldsymbol{w}(0))\|}$.

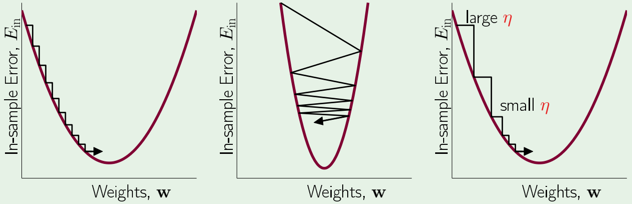
In other words, we update the weights as follows

$$\boldsymbol{w}(1) = \boldsymbol{w}(0) - \eta \frac{\nabla E_{\text{in}}(\boldsymbol{w}(0))}{\|\nabla E_{\text{in}}(\boldsymbol{w}(0))\|}.$$

Which step size?

Fixed-size step?

How η affects the algorithm:



η too small | η too large | variable η – just right

η should increase with the slope

η should be proportional to the length of the gradient

From step size to learning rate

Easy implementation

Instead of

$$\Delta \mathbf{w} = \eta\, \hat{\mathbf{v}}$$

$$= -\,\eta\, \frac{\nabla E_{\text{in}}(\mathbf{w}(0))}{\|\nabla E_{\text{in}}(\mathbf{w}(0))\|}$$

Have

$$\Delta \mathbf{w} = -\,\eta\ \nabla E_{\text{in}}(\mathbf{w}(0))$$

Fixed **learning rate** η

Gradient descent algorithm

Fixed learning rate gradient descent:

Initialize the weights at time step $t = 0$ to $\mathbf{w}(0)$.
for $t = 0, 1, 2, \ldots$ **do**
    Compute the gradient $\mathbf{g}_t = \nabla E_{\text{in}}(\mathbf{w}(t))$.
    Set the direction to move, $\mathbf{v}_t = -\mathbf{g}_t$.
    Update the weights: $\mathbf{w}(t+1) = \mathbf{w}(t) + \eta \mathbf{v}_t$.
    Iterate to the next step until it is time to stop.
Return the final weights.

Exercise

Exercise 3.7

For logistic regression, show that

$$\nabla E_{\text{in}}(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N}\frac{y_n\mathbf{x}_n}{1+e^{y_n\mathbf{w}^{\mathrm{T}}\mathbf{x}_n}}$$

$$= \frac{1}{N}\sum_{n=1}^{N}-y_n\mathbf{x}_n\theta(-y_n\mathbf{w}^{\mathrm{T}}\mathbf{x}_n).$$

Argue that a 'misclassified' example contributes more to the gradient than a correctly classified one.

Logistic regression algorithm

Logistic regression algorithm

Initialize the weights at $t = 0$ to $\mathbf{w}(0)$

for $t = 0, 1, 2, \ldots$ **do**

  Compute the gradient

$$\nabla E_{\text{in}} = -\frac{1}{N} \sum_{n=1}^{N} \frac{y_n \mathbf{x}_n}{1 + e^{y_n \mathbf{w}^{\intercal}(t)\mathbf{x}_n}}$$

  Update the weights: $\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \nabla E_{\text{in}}$

  Iterate to the next step until it is time to stop

Return the final weights $\mathbf{w}$

23/24

Stopping criterion

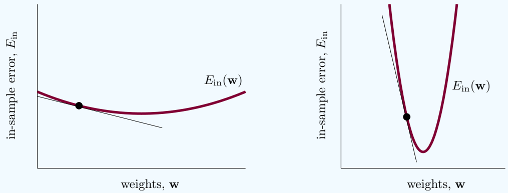
Typically the initial point $\boldsymbol{w}(0)$ is picked randomly, or we use prior knowledge about the problem. But when to stop the algorithm?

Some common choices (ϵ is a small prescribed threshold):

- $\|\nabla E_{\text{in}}(\boldsymbol{w}(t))\| < \epsilon$
- $|E_{\text{in}}(\boldsymbol{w}(t+1)) - E_{\text{in}}(\boldsymbol{w}(t))| < \epsilon$
- $\|\boldsymbol{w}(t+1) - \boldsymbol{w}(t)\| < \epsilon$
- $\frac{|E_{\text{in}}(\boldsymbol{w}(t+1)) - E_{\text{in}}(\boldsymbol{w}(t))|}{\max\{1, |E_{\text{in}}(\boldsymbol{w}(t))|\}} < \epsilon$
- $t > T$

Choosing the learning rate

- The size of the step taken in gradient descent, $-\eta \nabla E_{\text{in}}$, is proportional to the learning rate η.
- The optimal step size/learning rate depends on how wide or narrow the error surface is near the minimum.
- Wider surface $\implies$ we can take larger steps without overshooting. Since $\|\nabla E_{\text{in}}\|$ is small, we need a large η.



wide: use large η. narrow: use small η.

Choosing the learning rate

Variable Learning Rate Gradient Descent:

Initialize $\mathbf{w}(0)$, and η_0 at $t = 0$. Set $\alpha > 1$ and $\beta < 1$.
while stopping criterion has not been met **do**
  Let $\mathbf{g}(t) = \nabla E_{\text{in}}(\mathbf{w}(t))$, and set $\mathbf{v}(t) = -\mathbf{g}(t)$.
  if $E_{\text{in}}(\mathbf{w}(t) + \eta_t \mathbf{v}(t)) < E_{\text{in}}(\mathbf{w}(t))$ **then**
    accept: $\mathbf{w}(t+1) = \mathbf{w}(t) + \eta_t \mathbf{v}(t)$; $\eta_{t+1} = \alpha\eta_t$.
  else
    reject: $\mathbf{w}(t+1) = \mathbf{w}(t)$; $\eta_{t+1} = \beta\eta_t$.
  Iterate to the next step, $t \leftarrow t + 1$.

Steepest Descent (Gradient Descent + Line Search):

Initialize $\mathbf{w}(0)$ and set $t = 0$;
while stopping criterion has not been met **do**
  Let $\mathbf{g}(t) = \nabla E_{\text{in}}(\mathbf{w}(t))$, and set $\mathbf{v}(t) = -\mathbf{g}(t)$.
  Let $\eta^* = \text{argmin}_\eta\, E_{\text{in}}(\mathbf{w}(t) + \eta \mathbf{v}(t))$.
  $\mathbf{w}(t+1) = \mathbf{w}(t) + \eta^* \mathbf{v}(t)$.
  Iterate to the next step, $t \leftarrow t + 1$.

Other optimization methods

- Momentum, Nesterov Momentum, ...
- Adaptive learning rates: AdaGrad, RMSProp, RMS Prop, Adam
- Newtow’s Method, Conjugate gradient, BFGS, L-BFGS

Outline

About gradient descent

- Computing the full gradient is slow for big data
- Stuck at stationary points

Stochastic gradient descent

Stochastic gradient descent

GD minimizes:

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \underbrace{\mathrm{e}\left(h(\mathbf{x}_n), y_n\right)}_{\ln\left(1+e^{-y_n \mathbf{w}^{\mathsf{T}}\mathbf{x}_n}\right)} \quad \longleftarrow \text{ in logistic regression}$$

by iterative steps along $-\nabla E_{\text{in}}$:

$$\Delta \mathbf{w} = -\,\eta\,\nabla E_{\text{in}}(\mathbf{w})$$

∇E_{in} is based on all examples $(\mathbf{x}_n, y_n)$

"**batch**" GD

The stochastic aspect

Pick one $(\mathbf{x}_n, y_n)$ at a time. Apply GD to $\mathbf{e}\left(h(\mathbf{x}_n), y_n\right)$

"Average" direction:

$$\mathbb{E}_n\left[-\nabla \mathbf{e}\left(h(\mathbf{x}_n), y_n\right)\right] = \frac{1}{N}\sum_{n=1}^{N} -\nabla \mathbf{e}\left(h(\mathbf{x}_n), y_n\right)$$

$$= -\nabla\, E_{\text{in}}$$

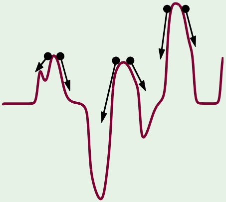
randomized version of GD

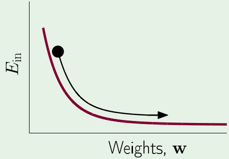
stochastic gradient descent (SGD)

Benefits of SGD

Benefits of SGD

1. cheaper computation
2. randomization
3. simple





Rule of thumb:

$\eta = 0.1$ works

randomization helps

Exercise

Exercise 3.10

(a) Define an error for a single data point $(\mathbf{x}_n, y_n)$ to be

$$\mathrm{e}_n(\mathbf{w}) = \max(0, -y_n \mathbf{w}^{\mathrm{T}} \mathbf{x}_n).$$

Argue that PLA can be viewed as SGD using e_n.

(b) For logistic regression with a very large $\mathbf{w}$, argue that minimizing E_{in} using SGD is similar to PLA. This is another indication that the logistic regression weights can be used as a good approximation for classification.

Mini-batch gradient descent

Compute the gradient using $1 \leq b \leq N$ samples.

1. Pick b examples ($1 \leq b \leq N$)
2. Apply batch GD to these b examples

Note: we can also shuffle the data and pick the mini-batches sequentially.

- $b = N$ is GD and $b = 1$ is SGD
- Bias and variance tradeoff
- A single pass through the entire training data is called an *epoch*. With mini-batches of size b, we update the parameters N/b times per epoch.
- We often need multiple epochs to obtain a good training accuracy.