Thursday · September 16, 2021

Stochastic Gradient Descent
Intro to Neural Networks

Yale

LING 380/780

Neural Network Models of Linguistic Structure

Gradient Descent

- Most models in this course will be too complicated to solve the minimization problem by hand.
- We will need a general algorithm to minimization the objective for complex architectures.

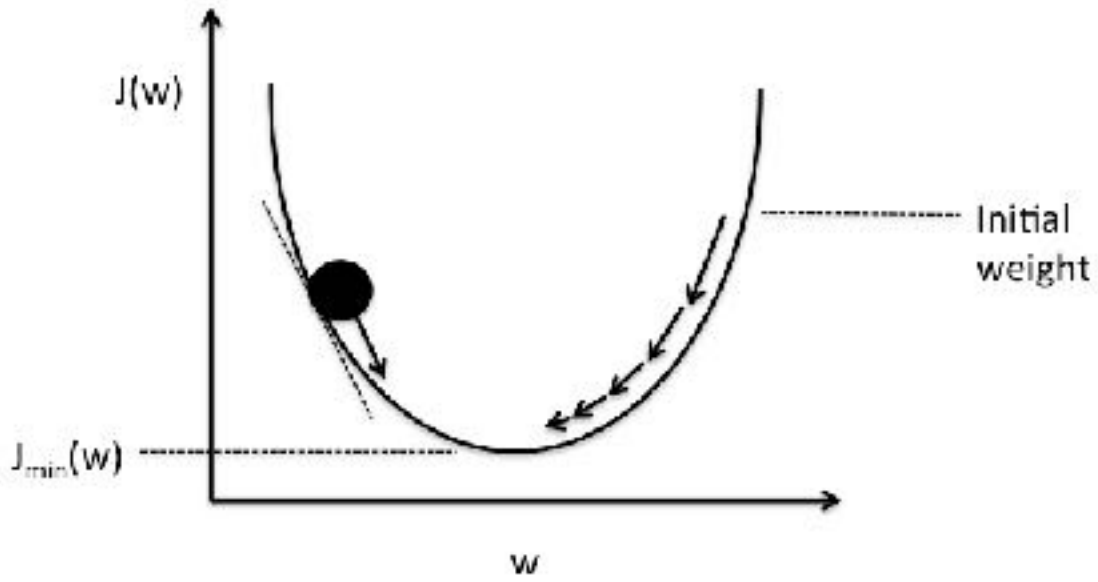
Gradient Descent

- Input: A dataset $\mathbb{D}$ and architecture $\hat{f}(\cdot; \boldsymbol{\theta})$
- Output: $\text{argmin}_{\boldsymbol{\theta}} \mathcal{L}$
- Initialize $\boldsymbol{\theta}$ to a random vector.
- Repeat indefinitely:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}$$

where η is the **learning rate**

Gradient descent



- Compute the derivative of the loss on training data with respect to the model parameters
- Update the model by taking a step in the direction away from the gradient:

$$w \leftarrow w - \eta \frac{\partial L(w, b)}{\partial w}$$

- Repeat until convergence (or for some fixed number of updates)

Stochastic gradient descent

- Instead of computing the gradient for the entire training set, we sample a minibatch B containing a fixed number of training examples, and take the average gradient:

$$w \leftarrow w - \frac{\eta}{|B|} \sum_{i \in B} \frac{\partial l^{(i)}(w, b)}{\partial w}$$

$$b \leftarrow b - \frac{\eta}{|B|} \sum_{i \in B} \frac{\partial l^{(i)}(w, b)}{\partial b}$$

Updating the gradients

Chain rule: $\frac{df(g(z))}{dx} = g'(z) \cdot f'(g(z))$

$$w \leftarrow w - \frac{\eta}{|B|} \sum_{i \in B} \frac{\partial (\hat{y}^{(i)} - y^{(i)})^2}{\partial w} \qquad b \leftarrow b - \frac{\eta}{|B|} \sum_{i \in B} \frac{\partial (\hat{y}^{(i)} - y^{(i)})^2}{\partial b}$$

$$w \leftarrow w - \frac{\eta}{|B|} \sum_{i \in B} \frac{\partial (wx^{(i)} + b - y^{(i)})^2}{\partial w} \qquad b \leftarrow b - \frac{\eta}{|B|} \sum_{i \in B} \frac{\partial (wx^{(i)} + b - y^{(i)})^2}{\partial b}$$

$$w \leftarrow w - \frac{\eta}{|B|} \sum_{i \in B} x \cdot 2(wx^{(i)} + b - y^{(i)}) \qquad b \leftarrow b - \frac{\eta}{|B|} \sum_{i \in B} 1 \cdot 2(wx^{(i)} + b - y^{(i)})$$

$$w \leftarrow w - \frac{2\eta}{|B|} \sum_{i \in B} x \cdot (\hat{y}^{(i)} - y^{(i)}) \qquad b \leftarrow b - \frac{2\eta}{|B|} \sum_{i \in B} (\hat{y}^{(i)} - y^{(i)})$$

$$\begin{bmatrix} w \\ b \end{bmatrix} = \begin{bmatrix} w \\ b \end{bmatrix} - \frac{\eta}{|B|} \sum_{i \in B} \begin{bmatrix} 2x^{(i)}(\hat{y}^{(i)} - y^{(i)}) \\ 2(\hat{y}^{(i)} - y^{(i)}) \end{bmatrix}$$

Another gradient update: SGNS

$$
\begin{aligned}
L_{CE} &= -\log\left[P(+|w,c_{pos})\prod_{i=1}^{k}P(-|w,c_{neg_i})\right] \\
&= -\left[\log P(+|w,c_{pos}) + \sum_{i=1}^{k}\log P(-|w,c_{neg_i})\right] \\
&= -\left[\log P(+|w,c_{pos}) + \sum_{i=1}^{k}\log\left(1-P(+|w,c_{neg_i})\right)\right] \\
&= -\left[\log \sigma(c_{pos}\cdot w) + \sum_{i=1}^{k}\log \sigma(-c_{neg_i}\cdot w)\right]
\end{aligned}
$$

Another gradient update: SGNS

$$\frac{\partial L_{CE}}{\partial c_{pos}} = \frac{\partial - [\log \sigma(c_{pos} \cdot w) + \sum_{i=1}^{k} \log \sigma(-c_{neg_i} \cdot w)]}{\partial c_{pos}}$$

$$= -\frac{1}{\sigma(c_{pos} \cdot w)} \frac{\partial \sigma(c_{pos} \cdot w)}{\partial c_{pos}}$$

$$\frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1 - \sigma(x))$$

$$= -\frac{1}{\sigma(c_{pos} \cdot w)} \sigma(c_{pos} \cdot w)(1 - \sigma(c_{pos} \cdot w))\ w$$

$$= [\sigma(c_{pos} \cdot w) - 1]w$$

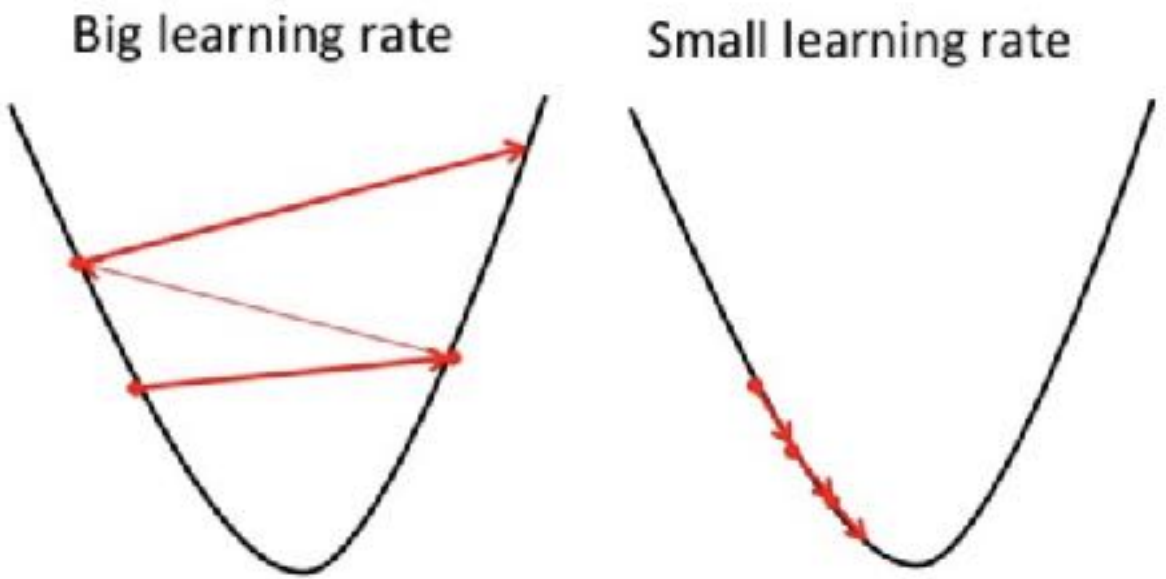
$$c_{pos}^{t+1} = c_{pos}^{t} - \eta[\sigma(c_{pos}^{t} \cdot w) - 1]w$$

Stochastic gradient descent

- Still need to fix **hyperparameters**
  - Learning rate η
  - Minibatch size $|B|$

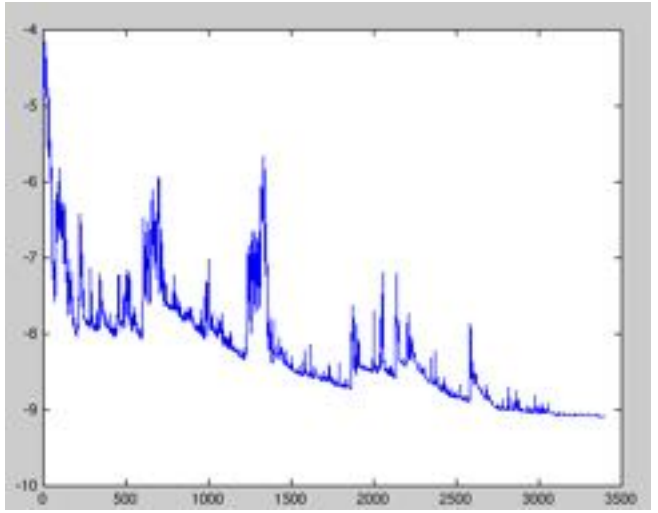
Choosing hyperparameters

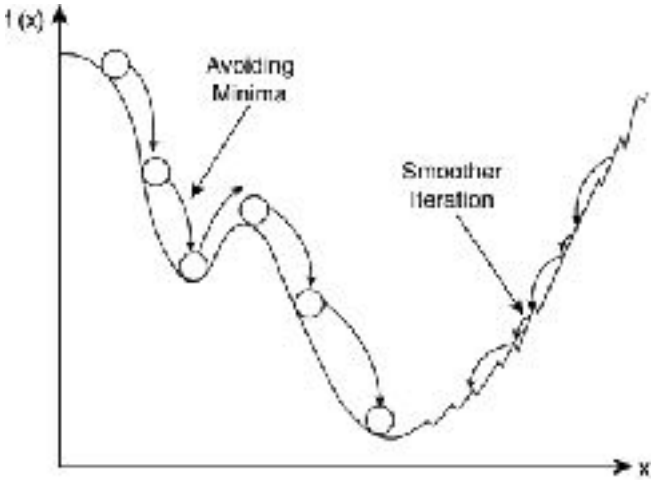
- **Learning rate** and the Goldilocks problem:



Choosing hyperparameters

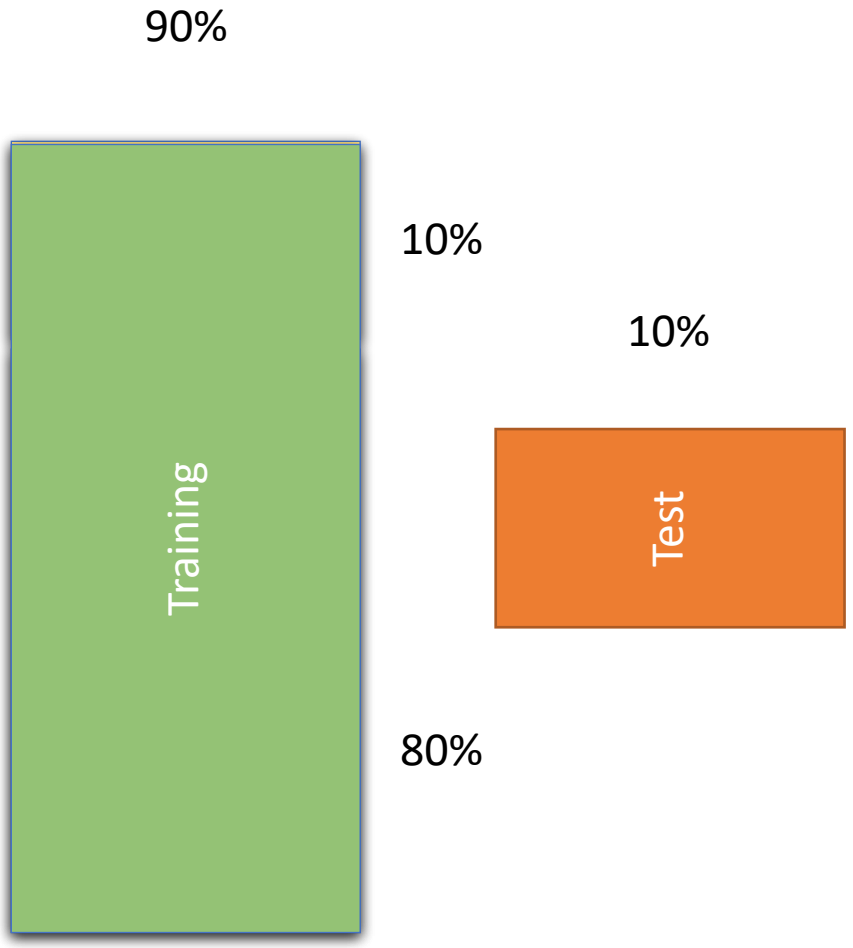
- **Batch size**: estimating the gradient from a sample of the training data means that we are not getting the true gradient.
- Such fluctuations can be useful for avoiding local minima of non-convex loss functions





Choosing hyperparameters

- Train-Dev-Test Split:
  - Divide “training” set into training and development/validation set.
  - For each hyperparameter choice h:
    - Train using h on training set
  - Choose hyperparameters that produce the best performance on the dev set
  - When training is complete, train the model on training set(with the best hyperparameters) and test the test set



Choosing hyperparameters

- k-fold Cross Validation:
  - Divide “training” set into k even “folds”.
  - For each hyper parameter choice h:
    - For each fold f:
      - Train using h on training data without f and evaluate on f
  - Choose hyperparameters that produce the best average performance across folds
  - When training is complete, train the model on training set (with the best hyperparameters) and test the test set