Applied Machine Learning

Deep Neural Networks and Backpropagation

Deep Neural Networks and Backpropagation

- Backpropagation Algorithm
- Training Deep Neural Networks

Backpropagation Algorithm

- Layer $i-1$: changes in Loss at output with respect to $\theta^{(i-1)}$
  - $\nabla_{\theta^{(i)}} L = \nabla_{\mathbf{o}^{(D)}} L \times \mathbf{J}_{\mathbf{o}^{(D)};\mathbf{u}^{(D)}} \times \ldots \mathbf{J}_{\mathbf{o}^{(i+1)};\mathbf{u}^{(i+1)}} \times \mathbf{J}_{\mathbf{o}^{(i)};\theta^{(i)}}$

$$
\begin{aligned}
\mathbf{v}^{(D)} &= \nabla_{\mathbf{o}^{(D)}} L \\
\nabla_{\theta^{(D)}} L &= \mathbf{v}^{(D)} \times \mathbf{J}_{\mathbf{o}^{(D)};\theta^{(D)}} \\
&\vdots \\
\mathbf{v}^{(i)} &= \mathbf{v}^{(i+1)} \times \mathbf{J}_{\mathbf{o}^{(i+1)};\mathbf{u}^{(i+1)}} \\
\nabla_{\theta^{(i)}} L &= \mathbf{v}^{(i)} \times \mathbf{J}_{\mathbf{o}^{(i)};\theta^{(i)}} \\
&\vdots
\end{aligned}
$$

- Backpropagation
  - Forward pass:
    - $\mathbf{u}^{(1)} = \mathbf{x}$
    - for each layer i from 1 to D:
      - $\mathbf{u}^{(i+1)} = \mathbf{o}^{(i)}(\mathbf{u}^{(i)}, \theta^{(i)})$
  - Backward pass:
    - $$\begin{aligned} \mathbf{v}^{(D)} &= \nabla_{\mathbf{o}^{(D)}} L \\ \nabla_{\theta^{(D)}} L &= \mathbf{v}^{(D)} \times \mathbf{J}_{\mathbf{o}^{(D)};\theta^{(D)}} \end{aligned}$$
    - for each layer i from D-1 to 1:
      - $$\begin{aligned} \mathbf{v}^{(i)} &= \mathbf{v}^{(i+1)} \times \mathbf{J}_{\mathbf{o}^{(i+1)};\mathbf{u}^{(i+1)}} \\ \nabla_{\theta^{(i)}} L &= \mathbf{v}^{(i)} \times \mathbf{J}_{\mathbf{o}^{(i)};\theta^{(i)}} \end{aligned}$$

Parameter Initialization

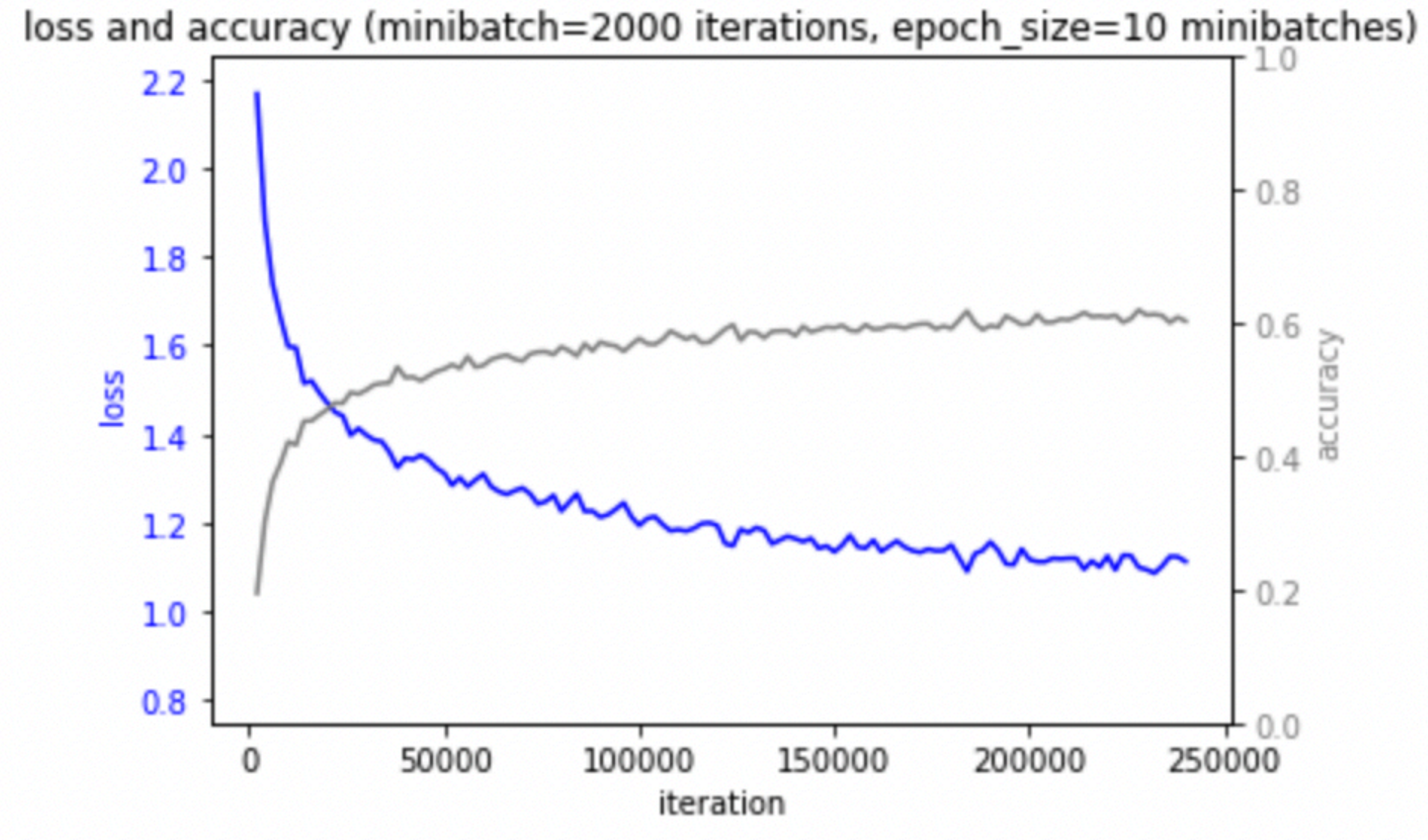
- Initial value of parameters
  - All 0 => bad option
  - Close to their optimal value
  - Each to a random value
    - from zero mean normal distribution with small standard deviation (e.g. 0.01)

Feature Preprocessing

- The relative distribution of different feature inputs may affect learning
    - Large variations in their standard deviations
        - Weights for inputs with larger standard deviations: larger
            - if stepsize is small, it takes longer to build them
        - Weights for inputs with smaller standard deviations: smaller
            - if stepsize is large, it's unlikely to find them
- Preprocessing:
    - Normalize each input to have zero mean and unit standard deviation
    - Apply domain knowledge

Training Loss and Accuracy

- Plots: Loss vs Epochs, Accuracy vs Epochs
- Stepsize or Learning Rate
  - smaller learning rate
    - smoother curves => gradual walk towards minimum
    - more likely to fall into non-optimal loss
  - larger learning rate
    - more noisy curves => jumps on the gradient
    - more likely to diverge
- Reducing loss does not necessarily improve accuracy



Data

- It is hard to succesfully train a Deep Neural Network
- Datasets
  - Many times, the more data the better, up to a point

Units

- Units
  - Redundant units
    - May result in cancellation of some at next layer
    - Dropout
      - before each training step
        - set outputs of some randomly selected units to zero
          - do not consider output units
  - Dead units
    - output 0 for every data item => Gradient 0 => dead unit
    - smaller learning rates help
    - many more units

Gradient Obfuscation

- Gradient Obfuscation
    - poor parameter estimates close to output layer => poor gradient update
    - the more layers above, the worst
- initialize with good estimate
- rescaling
- change connectivity structure

Training Deep Neural Networks

- Implementation
  - GPUs
  - APIs
    - describe layers, connectivity, gradients, unit functions
    - map onto the available computing resources (e.g., GPU)
    - training, and evaluation

Deep Neural Networks and Backpropagation

- Backpropagation Algorithm
- Training Deep Neural Networks

Applied Machine Learning

Deep Neural Networks and Backpropagation