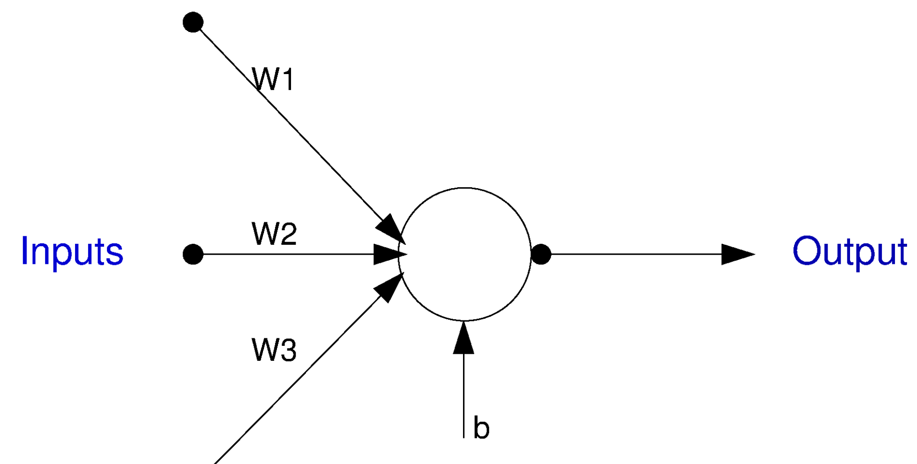
Applied Machine Learning

Training a Simple Neural Network

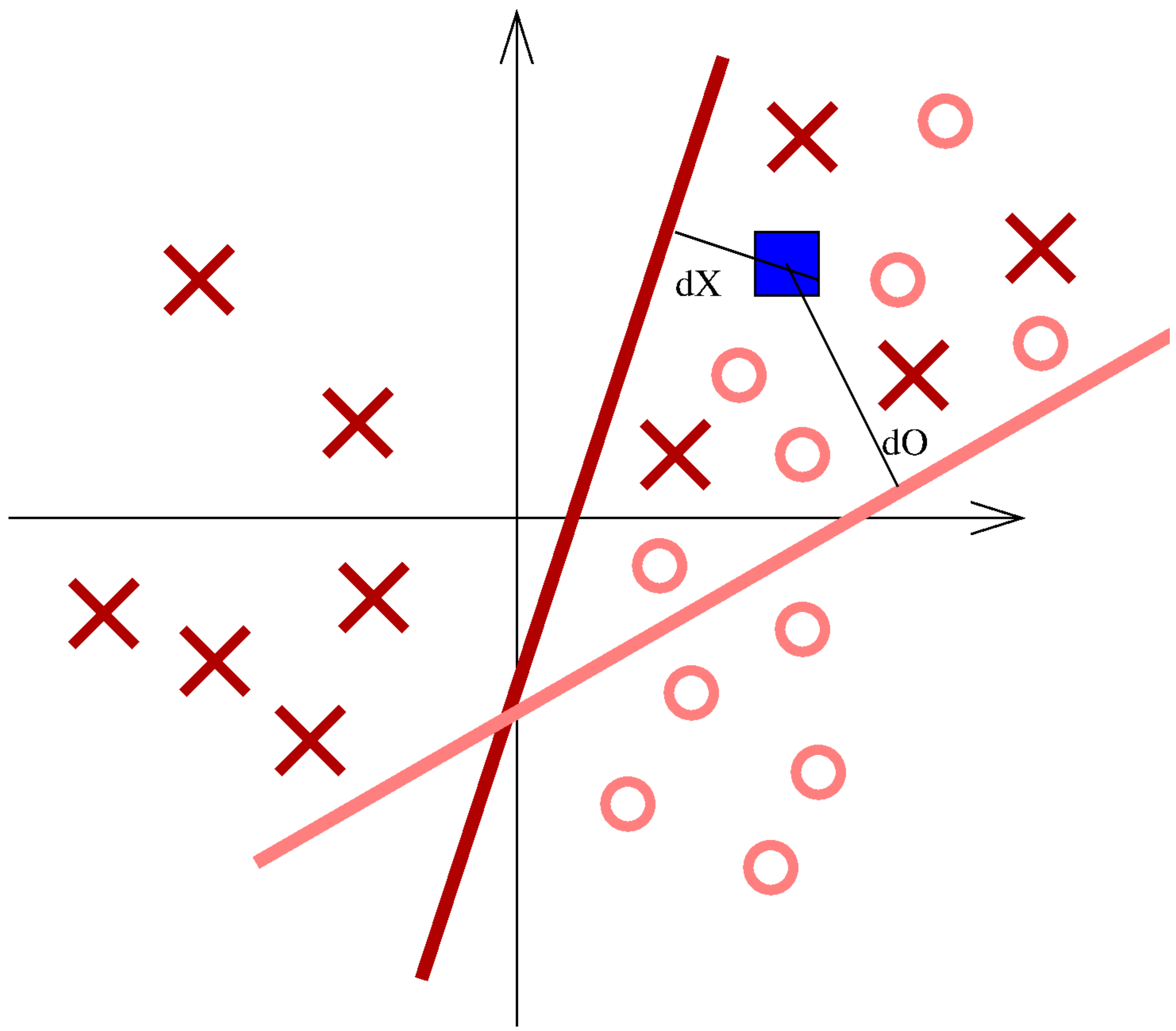
Training a Simple Neural Network

- Cost function for simple neural network
- Gradient
- Stochastic Gradient Descent
- Cross-Validation

Simple Neural Network



- Linear splitting: $u = \sum_i w_i x_i + b$.
- Rectified Linear Unit - **ReLU:** $F(u) = \max(0,u)$
- softmax$(u) = \mathbf{s}(\mathbf{o}) = \dfrac{1}{\sum_k e^{o_k}} \begin{bmatrix} e^{o_1} \\ \vdots \\ e^{o_C} \end{bmatrix}$
- Probability of input $\mathbf{x}_i$ corresponding to class j
  $p(y_j = 1 \mid \mathbf{x}_i, \mathbf{w}^{(j)}, b^{(j)}) = s_j(\mathbf{o}(\mathbf{x}_i, \mathbf{w}^{(j)}, b^{(j)}))$



Simple Neural Network: Cost function

- Goal: find the vectors of parameters for all the units:
  - $\theta^{(j)} = \begin{bmatrix} \mathbf{w}^{(j)} \\ b^{(j)} \end{bmatrix}$
- Cost function:
  - Loss: Maximize log likelihood of training data under probability model
  - Penalty: Minimize parameters
  - $S(\theta, x; \lambda) =$ loss $+ \lambda$ penalty

- Loss:
  - Minimize $-\log p(\mathbf{y}_i \,|\, \mathbf{x}_i, \theta)$
    - inputs $\mathbf{x}_i$, parameters θ, outputs $\mathbf{y}_i$, N items in dataset
  - Log Loss or Cross-Entropy Loss $\frac{1}{N}\sum_i [-\log p(\mathbf{y}_i \,|\, \mathbf{x}_i, \theta)] \quad =$
    - $= \quad \frac{1}{N}\sum_{i=1}^{N} [-\mathbf{y_i} \log \mathbf{s}(\mathbf{o}(\mathbf{x}_i, \theta))]$
  - Penalization: $\frac{1}{2}\sum_{i=1}^{C} \mathbf{w}^{(\mathbf{i})\top}\mathbf{w}^{(\mathbf{i})}$
  - Cost Function: $S(\theta, x; \lambda) \quad =$
    - $= \quad \frac{1}{N}\sum_{i=1}^{N} [-\mathbf{y_i} \log \mathbf{s}(\mathbf{o}(\mathbf{x}_i, \theta))] + \lambda \frac{1}{2}\sum_{i=1}^{C} \mathbf{w}^{(\mathbf{i})\top}\mathbf{w}^{(\mathbf{i})}$

Simple Neural Network: Training

- Goal: find the vectors of parameters θ that minimize the cost function
- Approach:
  - Stochastic Gradient Descent
    - It is hard to find the global minimum
    - reduce the loss over time
  - Minibatches: subset of dataset formed by selecting M items uniformly at random
  - Training
    - Iterate
      - form minibatch with M items from dataset selected uniformly at random
      - Apply Stochastic Gradient Descent to minibatch

Simple Neural Network: Gradient

- Cost Function: $S(\theta, x; \lambda) \quad = \quad \frac{1}{N} \sum_{i=1}^{N} [-\mathbf{y_i} \log \mathbf{s}(\mathbf{o}(\mathbf{x}_i, \theta))] + \lambda \frac{1}{2} \sum_{i=1}^{C} \mathbf{w^{(i)}}^\top \mathbf{w^{(i)}}$
- Penalty term for unit j: $\lambda \frac{1}{2} \mathbf{w^{(j)}}^\top \mathbf{w^{(j)}}$
- Gradient for the penalty term for unit j with respect to parameters
  - with respect to weight w_a in unit j: $\frac{\partial}{\partial w_a^j} \lambda \frac{1}{2} \mathbf{w^{(j)}}^\top \mathbf{w^{(j)}} = \lambda w_a^j$
  - with respect to bias in unit j: $\frac{\partial}{\partial b^{(j)}} \lambda \frac{1}{2} \mathbf{w^{(j)}}^\top \mathbf{w^{(j)}} = 0$

Simple Neural Network: Gradient

- Cost Function: $S(\theta, x; \lambda) \quad =$
  - $= \quad \frac{1}{N}\sum_{i=1}^{N}\left[-\mathbf{y_i}\log \mathbf{s}(\mathbf{o}(\mathbf{x}_i,\theta))\right] + \lambda\frac{1}{2}\sum_{i=1}^{C}\mathbf{w^{(i)}}^{\top}\mathbf{w^{(i)}}$
- loss term for data item i
  - $-\mathbf{y}_i \log \mathbf{s}(\mathbf{o}(\mathbf{x}_i,\theta)) \quad = \quad \sum_{u=1}^{C} y_u \log s_u(o(\mathbf{x}_i,\theta))$
  - loss term in unit u: $y_u \log s_u(o(\mathbf{x}_i,\theta))$
- Gradient for the loss term for data item i with respect to parameters
  - with respect to weight w_a in unit j
    - $\frac{\partial}{\partial w_a^{(j)}} y_u \log s_u(o(\mathbf{x}_i,\theta)) \quad = \quad y_u\left[\sum_v \frac{\partial \log s_u}{\partial o_v}\frac{\partial o_v}{\partial w_a^{(j)}}\right]$
  - with respect to bias in unit j
    - $\frac{\partial}{\partial b^{(j)}} y_u \log s_u(o(\mathbf{x}_i,\theta)) \quad = \quad y_u\left[\sum_v \frac{\partial \log s_u}{\partial o_v}\frac{\partial o_v}{\partial b^{(j)}}\right]$

- Indicator function: $\mathbb{I}(\text{condition}) = \begin{cases} 1 & \text{condition=True} \\ 0 & otherwise \end{cases}$
- $\frac{\partial \log s_v}{o_v}$ $\qquad s_v = \frac{e^{o_v}}{\sum_k e^{o_k}}$
- $\log s_v \quad = \quad \log\left(\frac{e^{o_v}}{\sum_k e^{o_k}}\right) \quad = \quad o_v - \log\sum_k e^{o_k}$
- $\frac{\partial \log s_v}{o_v} \quad = \quad \begin{cases} 1 - \frac{e^{o_v}}{\sum_k e^{o_k}} = 1 - s_v & \text{if } u = v \\ 0 - \frac{e^{o_v}}{\sum_k e^{o_k}} = 0 - s_v & \text{if } u \neq v \end{cases}$
- $\frac{\partial \log s_u}{\partial o_v} \quad = \quad \mathbb{I}_{u=v} - s_v$

Simple Neural Network: Gradient

- Cost Function: $S(\theta, x; \lambda) \quad =$
  - $= \quad \frac{1}{N}\sum_{i=1}^{N}[-\mathbf{y_i}\log \mathbf{s}(\mathbf{o}(\mathbf{x}_i,\theta))] + \lambda\frac{1}{2}\sum_{i=1}^{C}\mathbf{w^{(i)}}^\top\mathbf{w^{(i)}}$
- loss term for data item i
  - $-\mathbf{y}_i \log \mathbf{s}(\mathbf{o}(\mathbf{x}_i,\theta)) \quad = \quad -\sum_{u=1}^{C} y_u \log s_u(o(\mathbf{x}_i,\theta))$
  - loss term in unit u: $y_u \log s_u(o(\mathbf{x}_i,\theta))$
- Gradient for the loss term for data item i with respect to parameters
  - with respect to weight w_a in unit j
    - $\frac{\partial}{\partial w_a^{(j)}} y_u \log s_u(o(\mathbf{x}_i,\theta)) \quad = \quad y_u\left[\sum_v \frac{\partial \log s_u}{\partial o_v}\frac{\partial o_v}{\partial w_a^{(j)}}\right]$
  - with respect to bias in unit j
    - $\frac{\partial}{\partial b^{(j)}} y_u \log s_u(o(\mathbf{x}_i,\theta)) \quad = \quad y_u\left[\sum_v \frac{\partial \log s_u}{\partial o_v}\frac{\partial o_v}{\partial b^{(j)}}\right]$

- Indicator function:
  $\mathbb{1}_{\text{condition}} = \begin{cases} 1 & \text{condition=True} \\ 0 & otherwise \end{cases}$
- $\frac{\partial \log s_u}{\partial o_v} \quad = \quad \mathbb{1}_{u=v} - s_v$
- $\frac{\partial o_v}{\partial w_a^{(j)}} = x_a^{(j)}\mathbb{1}_{o_v>0}\mathbb{1}_{u=j}$
- $\frac{\partial o_v}{\partial b^{(j)}} = \mathbb{1}_{o_v>0}\mathbb{1}_{u=j}$

Simple Neural Network: Training

- Iterate
  - form minibatch with M items from dataset selected uniformly at random
  - Apply Stochastic Gradient Descent with data from minibatch
    - Iteration n
      - $\theta^{(n+1)} = \theta^{(n)} - \eta^{(n)} \nabla_\theta$cost
        - $\eta^{(n)}$:learning rate or step size
          - $e(n)$: epoch for step n
          - Options:
            - $\eta^{(n)} = \dfrac{1}{a + b \cdot e(n)}$
            - $\eta^{(n)} = \eta(\gamma)^{-e(n)} \qquad \gamma > 1$
  - Stopping criteria: based on number of epochs

- Regularization constant λ: cross-validation
- Try values of λ at different scales through cross-validation
  - Iterate
    - split dataset into:
      - held-out validation set and training set
    - train network with training set and selected λ
    - evaluate on held-out validation set
  - error is average of held-out errors
- select λ with smallest held-out error
- train using the whole dataset

Training a Simple Neural Network

- Cost function for simple neural network
- Gradient
- Stochastic Gradient Descent
- Cross-Validation

Applied Machine Learning

Training a Simple Neural Network