ECE 598NSG/498NSU
Deep Learning in Hardware
Fall 2020

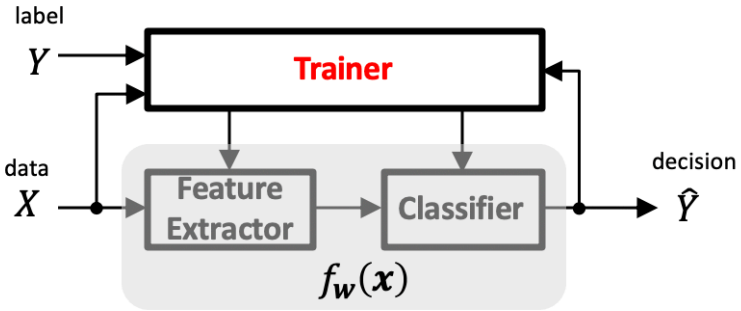
Training DNNs – SGD, Backprop and its Variants

Naresh Shanbhag
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign

http://shanbhag.ece.uiuc.edu

Today

- Overview of DNN training – the big picture
- The Stochastic Gradient Descent (SGD) training algorithm
- The back-prop algorithm

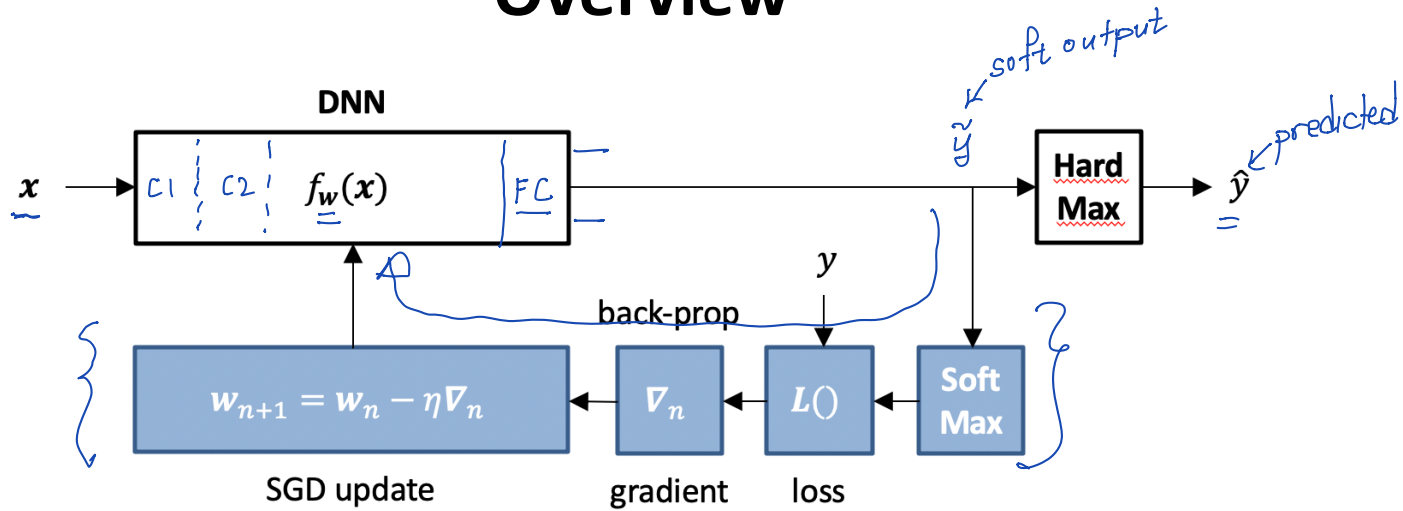


Supervised Training

- Sample: $z = (\boldsymbol{x}, y) = (\text{data}, \text{label})$;
- Training sample (example): samples used in training.
- Loss function: $L(\hat{y} = f_{\boldsymbol{w}}(\boldsymbol{x}), y)$; sample-specific value
- Family of functions parametrized by $\boldsymbol{w}$: $\mathcal{F}$: $f_{\boldsymbol{w}}(\boldsymbol{x}) = \hat{y}$
- Loss function evaluated on the training set: $Q(\boldsymbol{w})$
- learning (convergence) curves; learning rate; stability

DNN Training Set-up

Overview



- Forward pass – generates predicted label $\hat{y}$
- Backward pass – SGD-based back-prop to update weights

$f_w(x)$: multi-stage & non-linear.

Q: How to find w: we get accurate predictions?

Q: What is a measure of accuracy? → depends on the task.

Task → classification

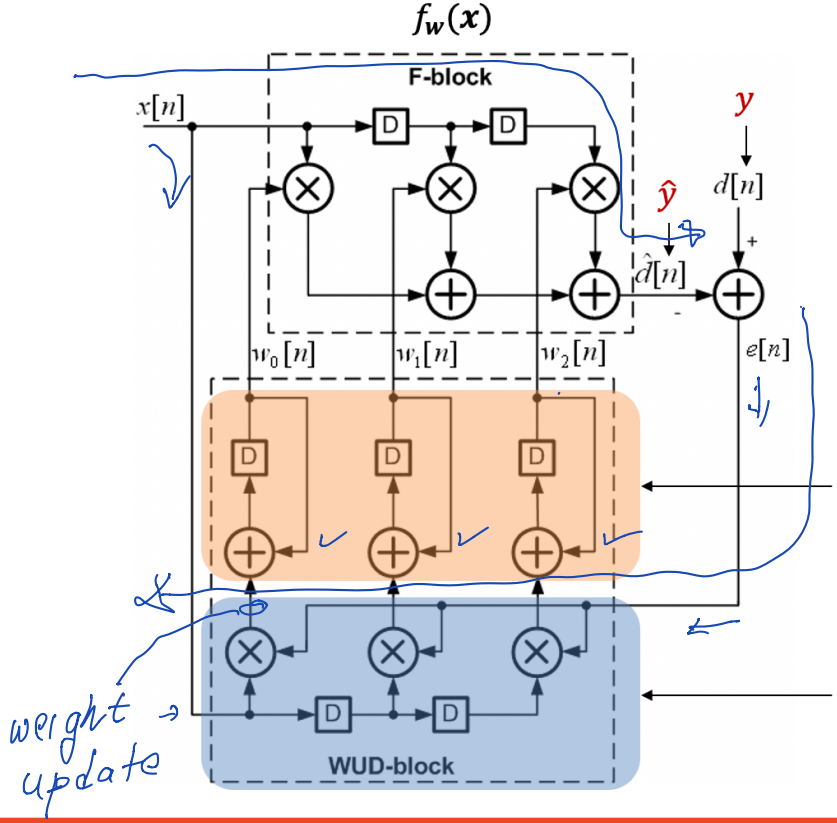
$P_e = Pr\{\hat{y} \neq y\}$: misclassification or error rate

(y: true label; $\hat{y}$: predicted label)

$1 - P_e$: accuracy

minimize P_e? How do we adjust w: P_e is minimized? → $\hat{y} \to y$

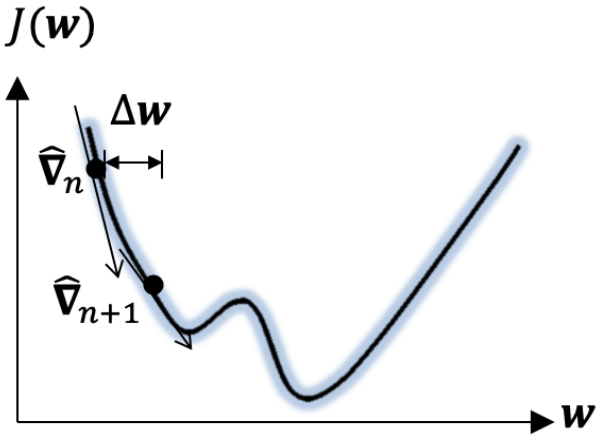
LMS via DNN Lens

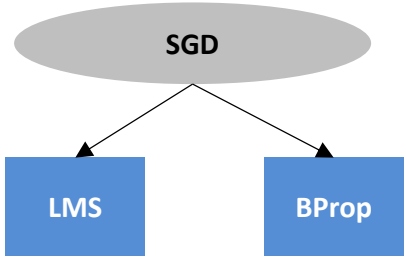


$$E[e^2[n]] = J(w)$$

$$\frac{\partial J(w)}{\partial w} \rightarrow 2e[n]x[n]$$

Stochastic Gradient Descent





- Calculate instantaneous gradient of loss function; update weight parameter by adding a small increment in the negative gradient

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n + \mu(-\hat{\boldsymbol{\nabla}}_n)$$

-ve of gradient of the loss fn. wrt. $\underline{w}$.

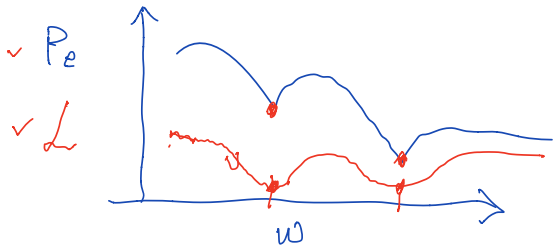
- $\hat{\nabla}_n = \dfrac{\partial \hat{J}(\boldsymbol{w}_n)}{\partial \boldsymbol{w}_n} \Rightarrow$ is the gradient of the *instantaneous loss function* wrt $\boldsymbol{w}_n$
- μ: step-size or learning rate

minimize P_e : directly is hard $\because$ $P_e(w)$

$$\frac{\partial P_e(w)}{\partial w} \longrightarrow \quad w_{n+1} = w_n + \mu\left(-\frac{\partial P_e(w)}{\partial w}\right)$$

Loss function : $\mathcal{L}(f_w(x), y) = \mathcal{L}(\hat{y}, y)$

↳ $\mathcal{L}$ is a surrgate/proxy for P_e



LMS: $MSE = \mathbb{E}\left[(\hat{y}-y)^2\right] \longrightarrow y, \hat{y} \in \mathbb{R}$

For DNNs : $\mathcal{L}$ is "cross entropy loss"

Entropy: of a RV X (discrete RV) → probability mass function $P_X(x)$

$$X \in \{-1, +1\} \rightarrow P_X(-1) = \frac{1}{2} \;;\; P_X(+1) = \frac{1}{2}$$

($-1 = x_1$, $+1 = x_2$)

Entropy of X : $H(X) = -\sum_i P_X(x_i) \log_2 P_X(x_i)$

$$H(X) = \sum_i P_X(x_i) \log_2\left(\frac{1}{P_X(x_i)}\right) = \mathbb{E}_{P_X}\left[\log_2\left(\frac{1}{P_X(x_i)}\right)\right]$$

↳ min. # of bits on average needed to represent X.

# of bits needed to rep. x_i

Cross-entropy: $X, Y \in \{-1, +1\}$, $X \sim P_X$, $Y \sim Q_X$

$$H(X,Y) = -\sum_i P_X(x_i) \log_2 (Q_X(x_i))$$

$$= +\sum_i P_X(x_i) \log_2 \left(\frac{1}{Q_X(x_i)}\right)$$

bits assigned

average # of bits needed to represent X when we assume Q_X as its distribution.

$H(X,Y) \geq H(X)$

In DNNs: $P_X \rightarrow$ true distribution of labels

$Q_X \rightarrow$ distribution generated network

Update $w \rightarrow$ nudge Q_X towards P_X, $\rightarrow$ "gap" or "distance" between Q_X & P_X is minimized.

↓

Kullback–Leibler divergence

$$D_{KL}(P_X \| Q_X) = \mathbb{E}_{P_X}\left(\log_2 \frac{P_X}{Q_X}\right)$$

$$= \sum_i P_X(x_i) \log_2 \frac{P_X(x_i)}{Q_X(x_i)}$$

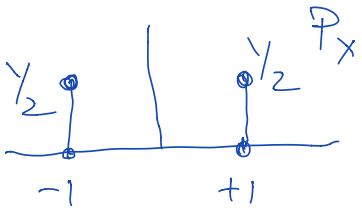
$$= \sum_i P_X \log_2 P_X - \sum_i P_X \log_2 Q_X = -H(X) + H(X,Y)$$

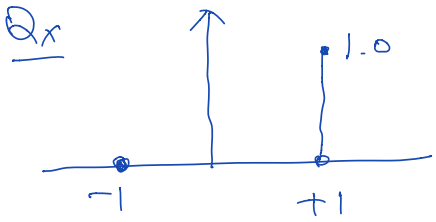
$$D_{KL}(P_x \| Q_x) = -H(X) + H(X,Y)$$

$$\min D_{KL}(P_x \| Q_x) = \min H(X,Y) \rightarrow \text{minimize CE loss.}$$

$$H(X,Y) \geq H(X) \rightarrow D_{KL}(P_x \| Q_x) \geq 0$$

Example: $x \in \{-1, +1\}$



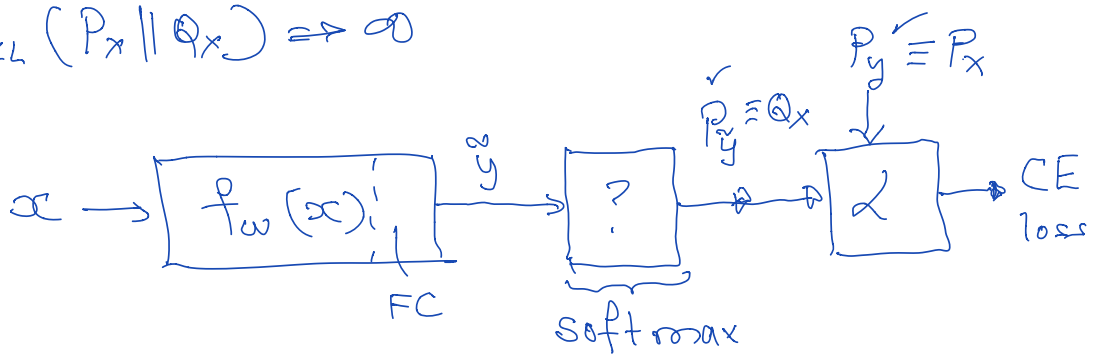


$$H(X) = -\left(\frac{1}{2} \times \log_2\left(\frac{1}{2}\right) + \frac{1}{2} \times \log_2\left(\frac{1}{2}\right)\right) = 1$$

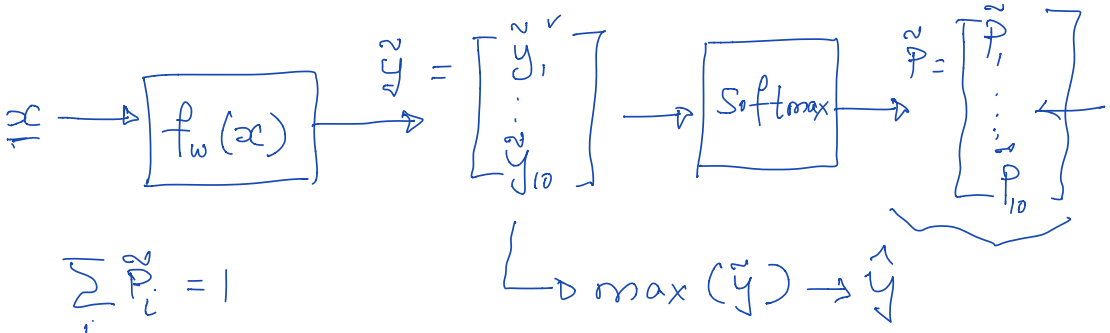
$$H(X,Y) = -\left(\frac{1}{2} \times \log_2(0) + \frac{1}{2} \times \log_2(1)\right) \rightarrow \infty$$

($\log_2(0) \rightarrow -\infty$, $\log_2(1) \rightarrow 0$)

$$D_{KL}(P_x \| Q_x) \Rightarrow \infty$$



DNN Example: 10-way classifier



$$\sum_i \tilde{p}_i = 1$$

$$\tilde{p}_i = \frac{e^{\tilde{y}_i}}{\sum_i e^{\tilde{y}_i}}$$ → softmax → generates the predicted distribution of class labels.

$\underline{p} = [0, 1, 0, \dots]$ — one-hot encoded vector

class 2 is true label.

$$\mathcal{L}(\hat{y}, y) = -\log_2 \tilde{p}_i \quad : \; i \leftarrow \text{true class}$$

$$= -\sum_i p_i \log_2 \tilde{p}_i$$

(p_i: 1-hot)

$$\frac{\partial(-\log_2 \tilde{p}_i)}{\partial \tilde{p}_i} = -\frac{1}{\tilde{p}_i \ln 2}$$

Ex: binary classifier

$$\tilde{y}_1 = w_1 x_1 + w_2 x_2 \; ; \quad \tilde{y}_2 = w_3 x_1 + w_4 x_2$$

True class = 1

$$\mathcal{L}(\hat{y}, y) = -\log_2 \tilde{p}_1$$

$$\tilde{p}_1 = \frac{e^{\tilde{y}_1}}{e^{\tilde{y}_1} + e^{\tilde{y}_2}}$$

$$\left(\frac{\partial \mathcal{L}()}{\partial w_1}\right) = ? \quad \text{— using chain rule}$$