Course Materials for Deep Generative Models

Northeastern University

These materials have been prepared and sourced for the course **Deep Generative Models** at Northeastern University. Every effort has been made to provide proper citations and credit for all referenced works.

If you believe any material has been inadequately cited or requires correction, please contact me at:

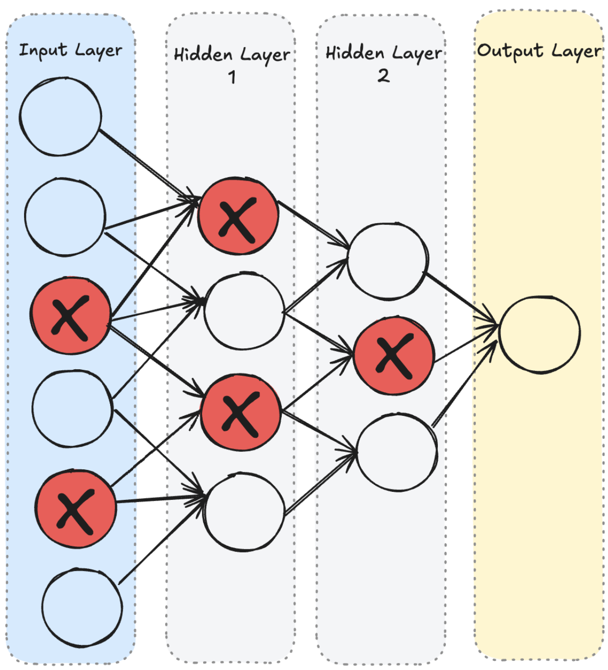
Instructor: Ramin Mohammadi
r.mohammadi@northeastern.edu

Thank you for your understanding and collaboration.

Dropout

Introduction

Dropout is a regularization technique used in neural networks to prevent overfitting. It works by randomly "dropping out" (setting to zero) a subset of neurons during each forward pass of training. This forces the network to learn redundant data representations, improving its generalization performance.



Thinned network: A neural network consisting of all units that survived dropout.
A neural network with n units can create a collection of 2^n thinned networks. For each node, we have either drop or not(binary). For example, if you have 10 fingers, you can count from 000000000 or 111111111 or from 0 to $2^n - 1 = 1024$.

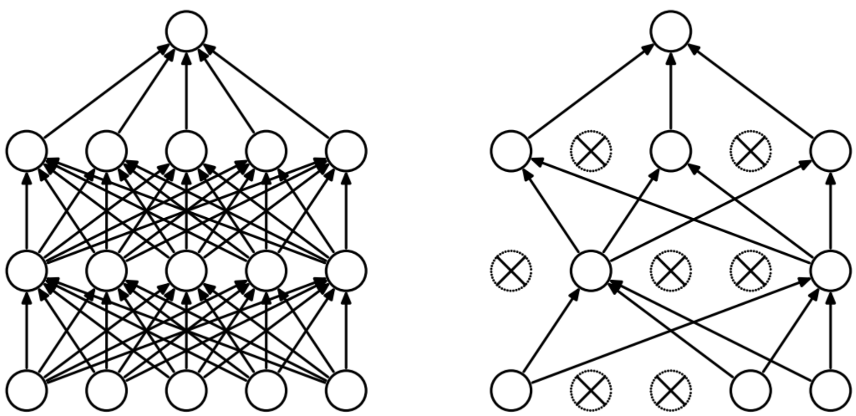
- **During training**, dropout samples from the number of thinned networks.
- **During testing**, we approximate the effect of averaging the predictions of all these thinned networks.

Consider a neural network with:

- L hidden layers,
- Z^i = input to layer i
- a^i = output to layer i
- w^i,b^i are weights and biases of layer i
- R_j^P = Bernouli value for node j in layer P

$$
\begin{aligned}
R_j^P &\approx \text{Bernouli(P)} \\
r^p &= \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}_{S_P} \\
\tilde{a}^P &= r^P \odot a^P \\
z_i^{P+1} &= w_i^{P+1} \tilde{a}^P + b_i^{P+1} \\
\tilde{a}^{P+1} &= f(z_i^{P+1})
\end{aligned}
$$

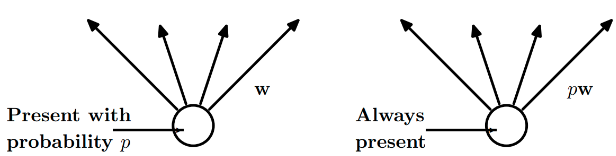
We only back-propagate on each thinned network.



(a) Standard Neural Net (b) After applying dropout.

Test Case

- use a single NN without dropout.
- Multiply all weights and biases by P(Bernoulli Parameter) because the weight/bias wasn't always available, so to ensure during the inference, that's also the case.
- every weight was used during the training with probability of P(Bernoulli) and was dropped with probability of (1 - p).

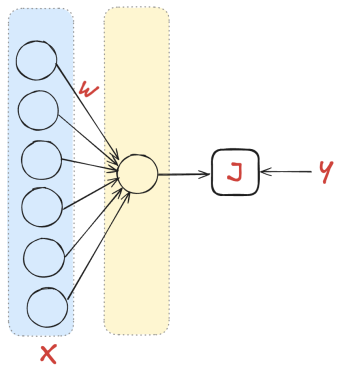


(a) At training time (b) At test time

Left: A unit at training time that is present with probability p and is connected to units in the next layer with weights w. Right: At test time, the unit is always present and the weights are multiplied by p. The output at test time is same as the expected output at training time.

- w_i^L, out of 10 epochs maybe, x times was used in training, we multiply $\frac{x}{10} * w_i^L$, by doing so we are getting on average results from 10 NNs. Technically each iteration we had a new NN as they were different in nodes. (like bagging)

Applying dropout to linear regression



Given,

$$X \in R^{m\text{x}d}; w \in R^{d\text{x}I}; y \in R^m$$

In Linear regression our goal is to minimize:

$$J = \|\mathbf{y} - X\mathbf{w}\|^2$$

$$R \in \mathbb{R}^{m\times d}, \quad R = \begin{bmatrix} R_{1,1} & R_{1,2} & \dots & R_{1,d} \\ R_{2,1} & R_{2,2} & \dots & R_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ R_{m,1} & R_{m,2} & \dots & R_{m,d} \end{bmatrix},$$

where each element $R_{i,j} \sim \text{Bernoulli}(p)$, i.e.,

$$R_{i,j} = \begin{cases} 1, & \text{with probability } p, \\ 0, & \text{with probability } 1-p. \end{cases}$$

our objective function becomes:

$$\min_{\mathbf{w}} \mathbb{E}_{R\sim\text{Bernoulli}(p)} \left[\|\mathbf{y} - (R * X)\mathbf{w}\|^2 \right]$$

$|y - Xw|^2$

$\Rightarrow$we apply dropout using Bernoulli p

$\Rightarrow |y - \underline{R \odot X}w|^2$

applying Bernoulli to x, so some of the weights will drop

$\Rightarrow |y - Mw|^2$

the goal is to understand the expectation of $R \odot X$

In each iteration R changes(Bernoulli$\approx$p), so some w will drop.

We want to see what is the expected value of w_i^L if we drop/keep it randomly after n epochs.

$= (y - Mw)^T(y - Mw) = y^Ty - y^Tmw - w^Tm^Ty + w^Tm^Tmw$

$= y^Ty - 2w^Tm^Ty + w^Tm^Tmw$

$\Rightarrow$we want to find:

$E[(y - Mw)^2]$ w.r.t. R

$\Rightarrow E[y^Ty - 2w^Tm^Ty + w^Tm^Tmw]$

$= E[y^Ty] - 2w^T\underline{(E[m])^T}y + w^TE[m^Tm]w$

break down:

$$E[\mathbf{M}] = \begin{bmatrix} E[M_{11}] & E[M_{12}] & \cdots & E[M_{1d}] \\ E[M_{21}] & E[M_{22}] & \cdots & E[M_{2d}] \\ \vdots & \vdots & \ddots & \vdots \\ E[M_{m1}] & E[M_{m2}] & \cdots & E[M_{md}] \end{bmatrix} \Rightarrow E[M_{ij}] = p \cdot X_{ij}$$

$\Rightarrow E[R \odot X] = E[M] = p \cdot X$

$$\Rightarrow E[(M^TM)_{ij}] = \sum_k E(R_{ki}R_{kj})X_{ki}X_{kj}$$

where $E[R_{ki}R_{kj}] = \begin{cases} \text{if i} \neq \text{j; } p^2 \\ \text{if i=j; } p \end{cases}$

Considering the above

$\Rightarrow E[y^Ty] - 2w^T(E[m])^Ty + w^TE[m^Tm]w = y^Ty - 2Pw^TX^Ty + w^TE[m^Tm]w$

We could solved it by directly multiplying P by X,

$$|y - pXw|^2 = y^Ty - 2pw^TX^Ty + p^2w^TX^TXw$$

So we can express:

$$\left[\|\mathbf{y} - (R * X)\mathbf{w}\|^2\right] = |y - pXw|^2 - p^2w^TX^TXw + w^TE[m^Tm]w$$

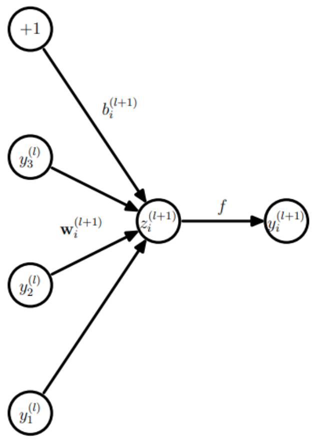
$$\Rightarrow |y - pXw|^2 + w^T[-p^2X^TX + E[m^Tm]]w$$

$$\text{considering that:} \begin{cases} \text{if i} \neq \text{j; } E[m^Tm] = p^2X^2 \\ \text{if i=j; } E[m^Tm] = p.\text{diagonal}(X^2) \end{cases}$$

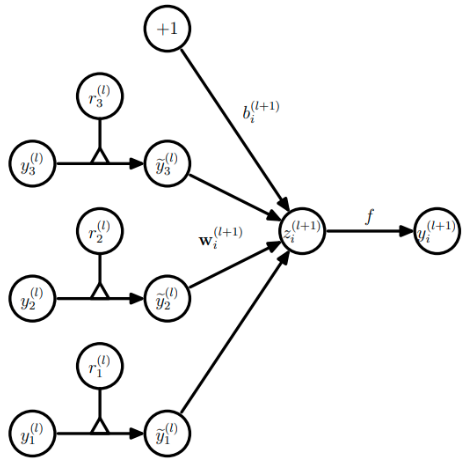
$$\Rightarrow |y - pXw|^2 + w^T[-p^2X^TX + pX^2]w$$

$$\Rightarrow |y - pXw|^2 + w^T[p * (1 - p)\text{diagonal}(X^2)]w$$

In general applying the dropout to linear regression is similar to Ridge Regression.



(a) Standard network



(b) Dropout network