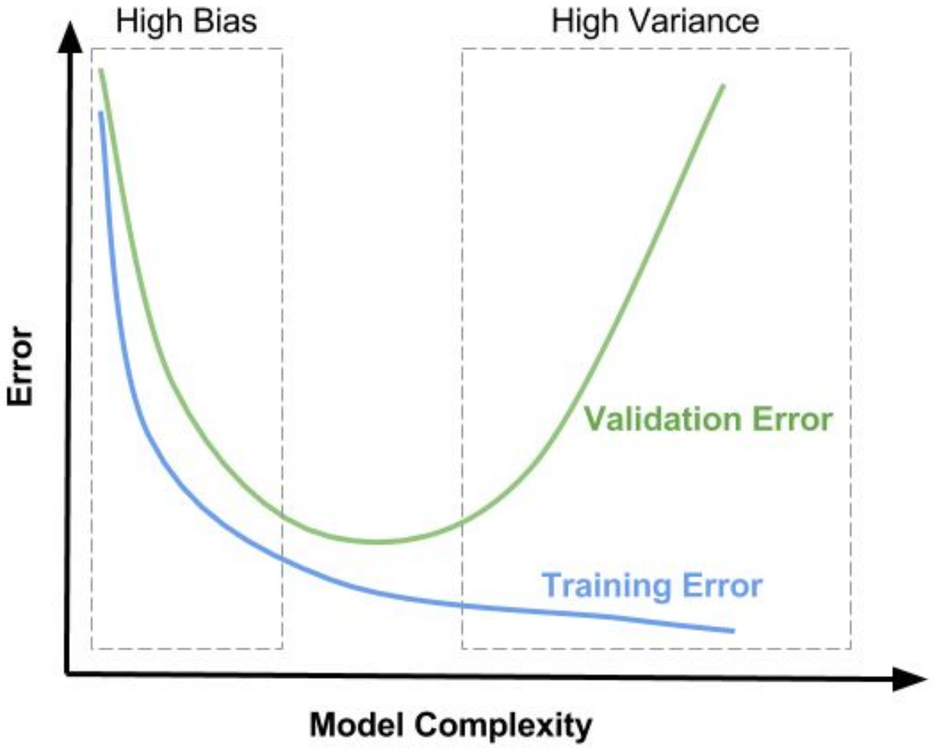
Neural Networks and Optimization IV

Dr. Parlett-Pelleriti

Bias Variance Tradeoff



Regularization

With great complexity comes great regularization

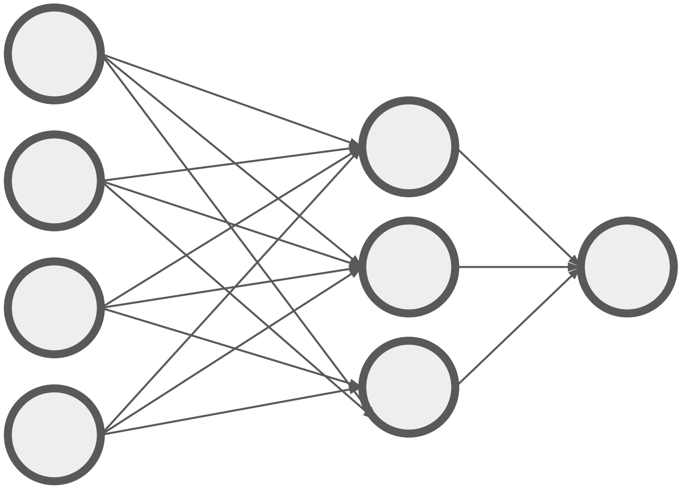
Regularization

- Penalization
- Bagging
- Dropout
- Early Stopping
- Batch Normalization

Penalization

- Best for smaller networks
- Added to EACH layer
- L1, L2, L1 + L2

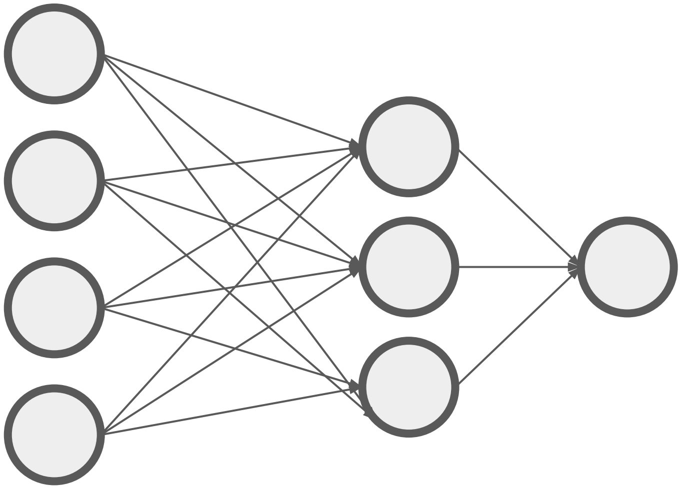
Penalization



Penalization

$$\text{penalized loss } = \text{loss } + \lambda\Omega(\theta)$$

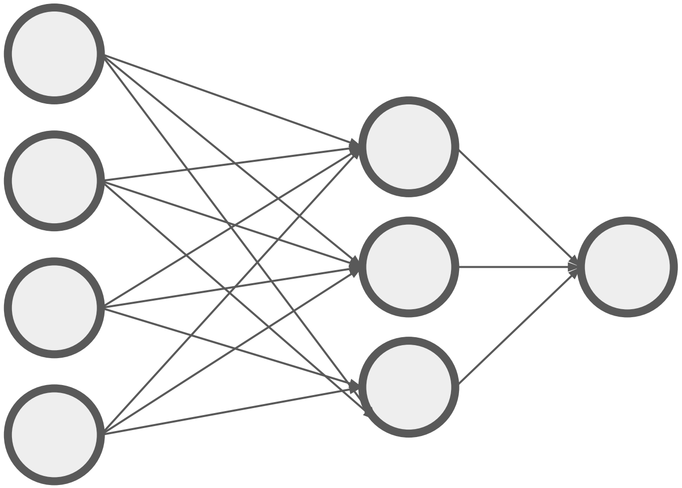
$\Omega(\theta)$ is some penalty on the size of the parameters θ



Penalization

$$\text{penalized loss} = \text{loss} + \lambda\Omega(\theta)$$

$\Omega(\theta)$ is some penalty on the size of the parameters θ



Typically we only penalize the weights, not the biases in a network

Penalization

$$\text{penalized loss } = \text{loss } + \lambda\Omega(\theta)$$

$\Omega(\theta)$ is some penalty on the size of the parameters θ

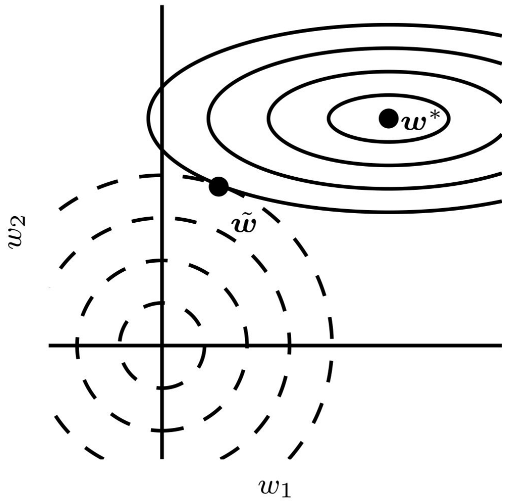


Figure 7.1

L_2 Penalization

$$\text{penalized loss} = \text{loss} + \lambda\Omega(\theta)$$

$\Omega(\theta)$ is some penalty on the size of the parameters θ

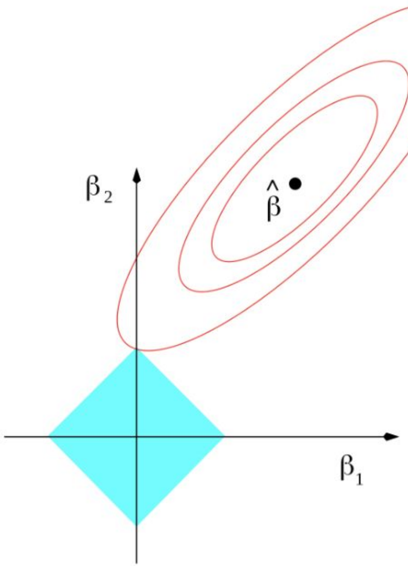
$$w_t = \underbrace{w_{t-1}}_{\text{old weights}} - \underbrace{\alpha * g_t}_{\text{update weights}}$$

$$w_t = \underbrace{(1 - \lambda\alpha)w_{t-1}}_{\text{shrink weights}} - \underbrace{\alpha * g_t}_{\text{update weights}}$$

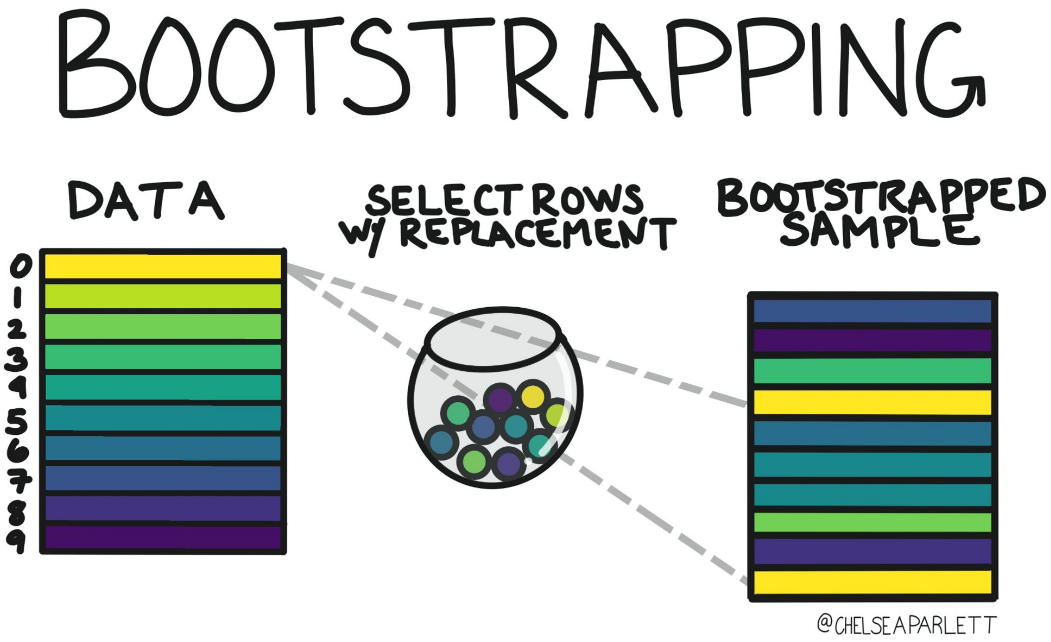
L_1 Penalization

$$\text{penalized loss} \ = \text{loss} \ + \lambda\Omega(\theta)$$

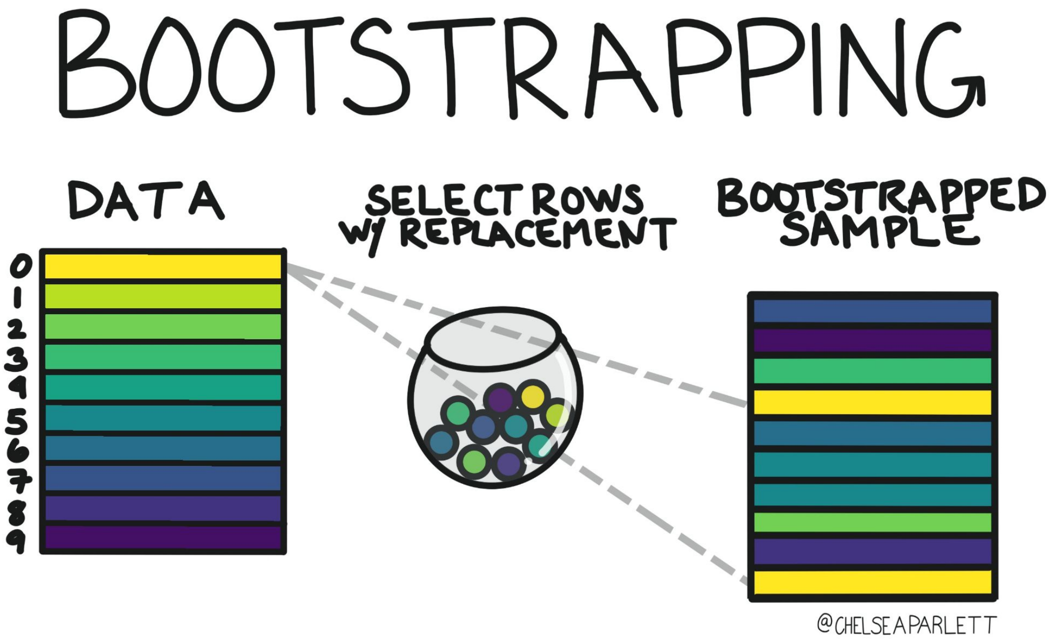
$\Omega(\theta)$ is some penalty on the size of the parameters θ



Bagging


BOOTSTRAPPING
DATA
0
1
2
3
4
5
6
7
8
9
SELECT ROWS
W/ REPLACEMENT
BOOTSTRAPPED
SAMPLE
@CHELSEAPARLETT

Bagging



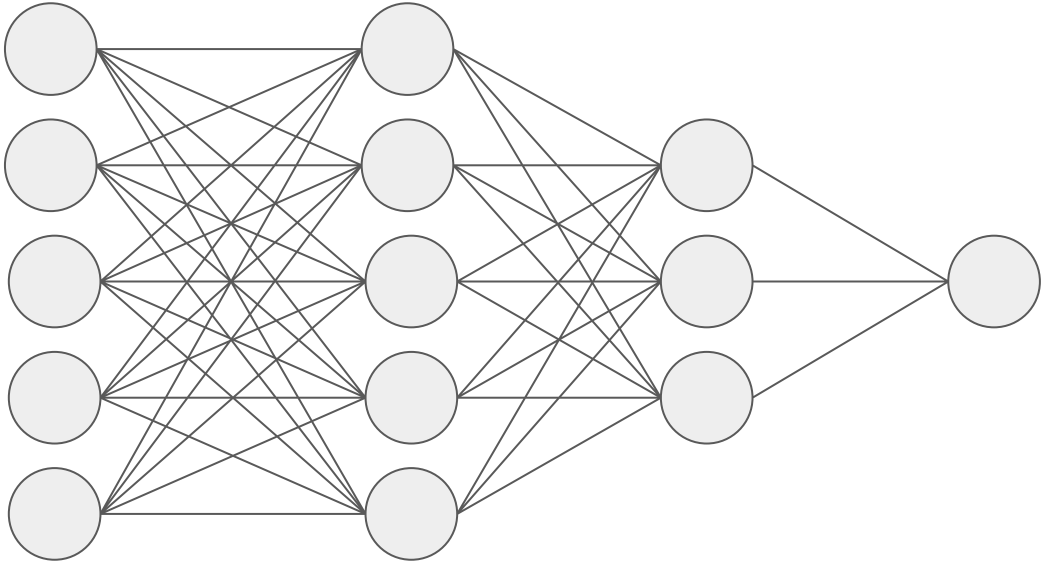
BOOTSTRAPPING
DATA
SELECT ROWS
W/ REPLACEMENT
BOOTSTRAPPED
SAMPLE
0
1
2
3
4
5
6
7
8
9
@CHELSEAPARLETT

*

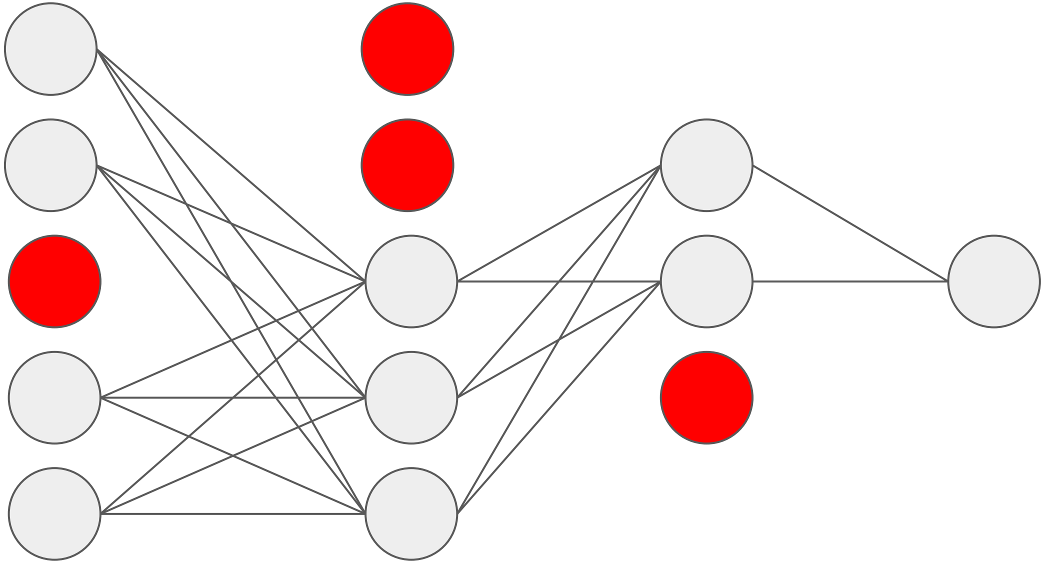
Dropout

- Comparison to Random Forest Bagging/Feature Selection
- Applied to each layer
- Network can’t over-rely on some connections

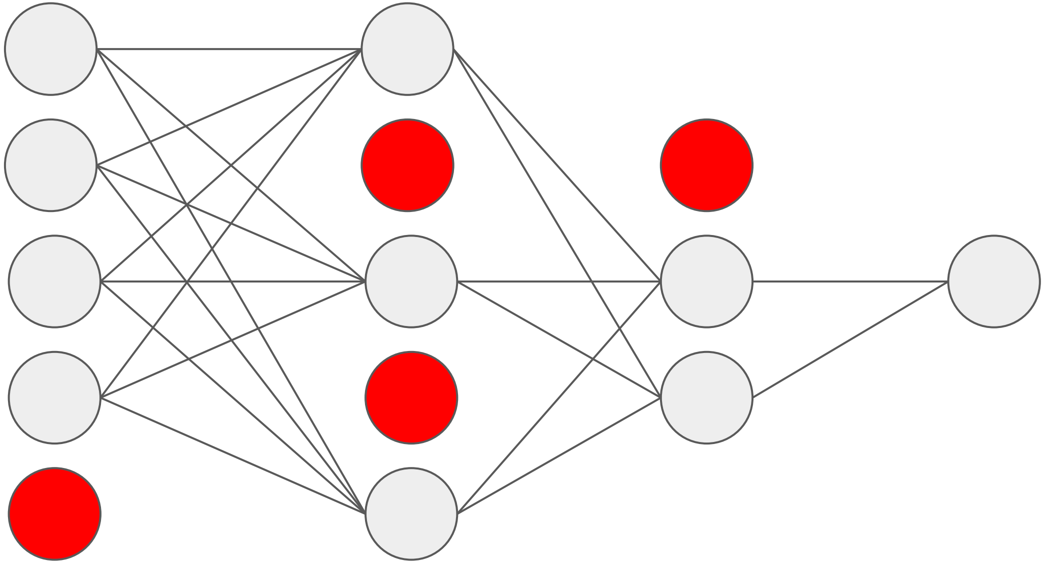
Dropout



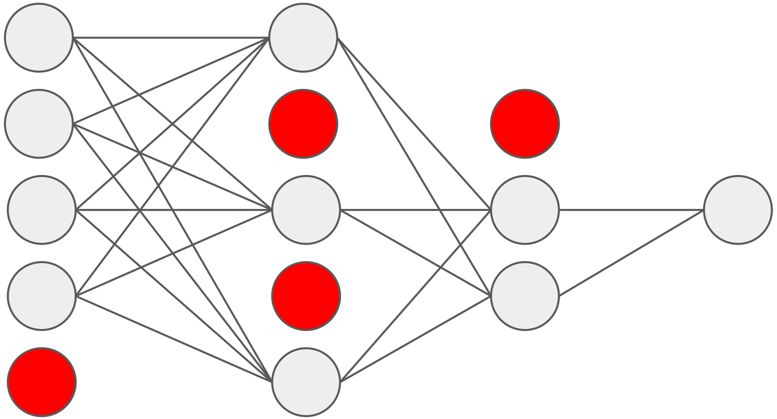
Dropout



Dropout



Dropout

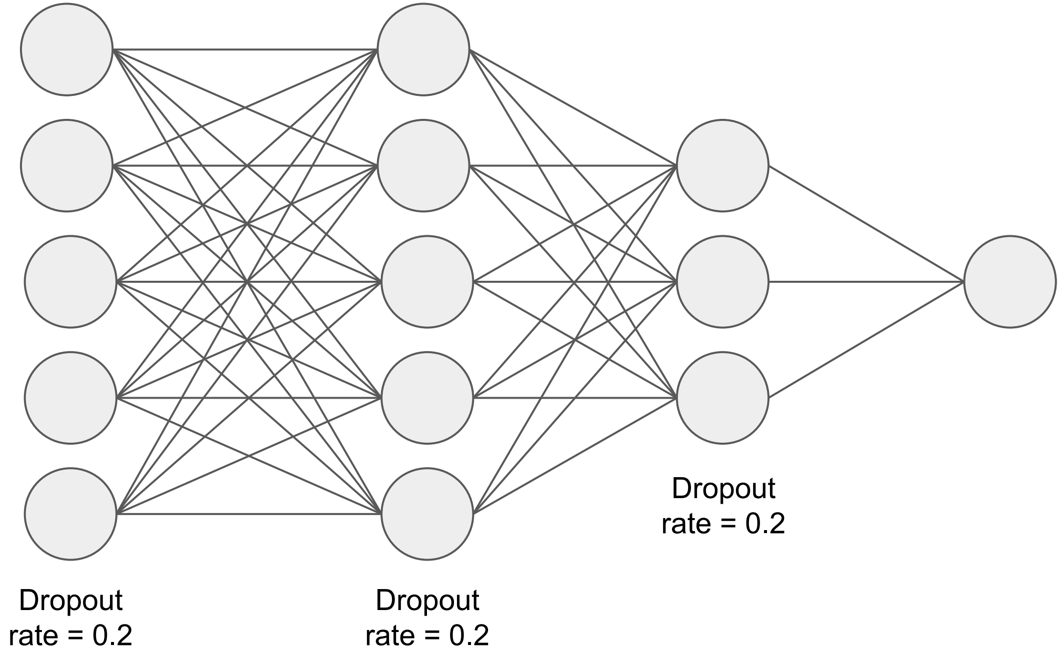


$$\begin{pmatrix} 0.1 & 0.4 & 0.2 & 0.3 & 0.2 \end{pmatrix}$$

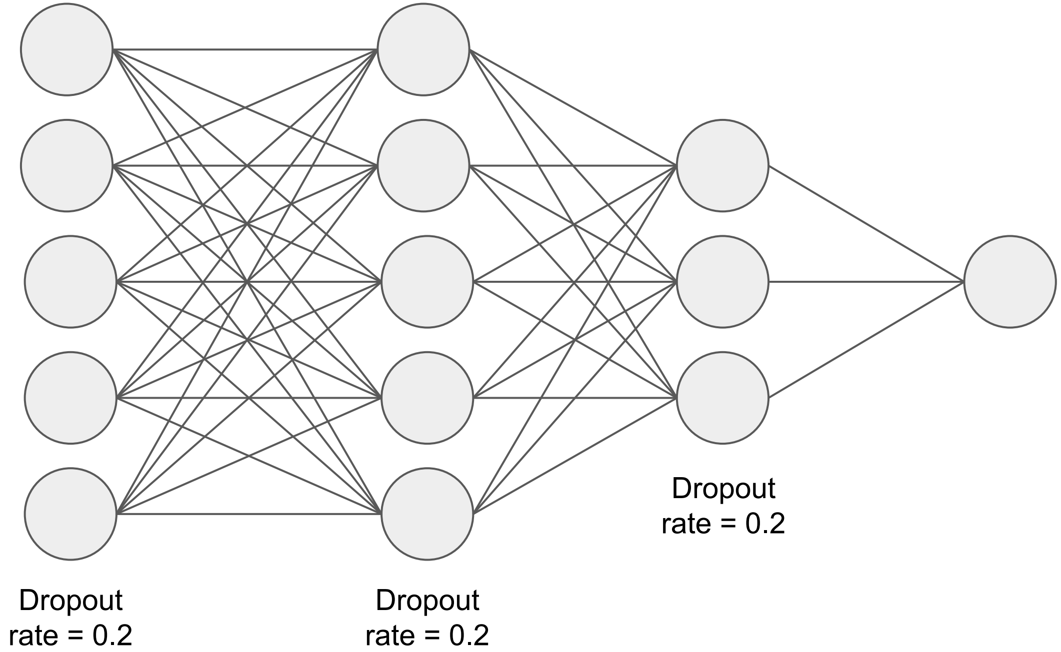
$$\begin{pmatrix} 0.1 & \color{red}{\mathbf{0}} & 0.2 & 0.3 & 0.2 \end{pmatrix}$$

$$\begin{pmatrix} 0.125 & \color{red}{\mathbf{0}} & 0.25 & 0.375 & 0.25 \end{pmatrix}$$

Dropout

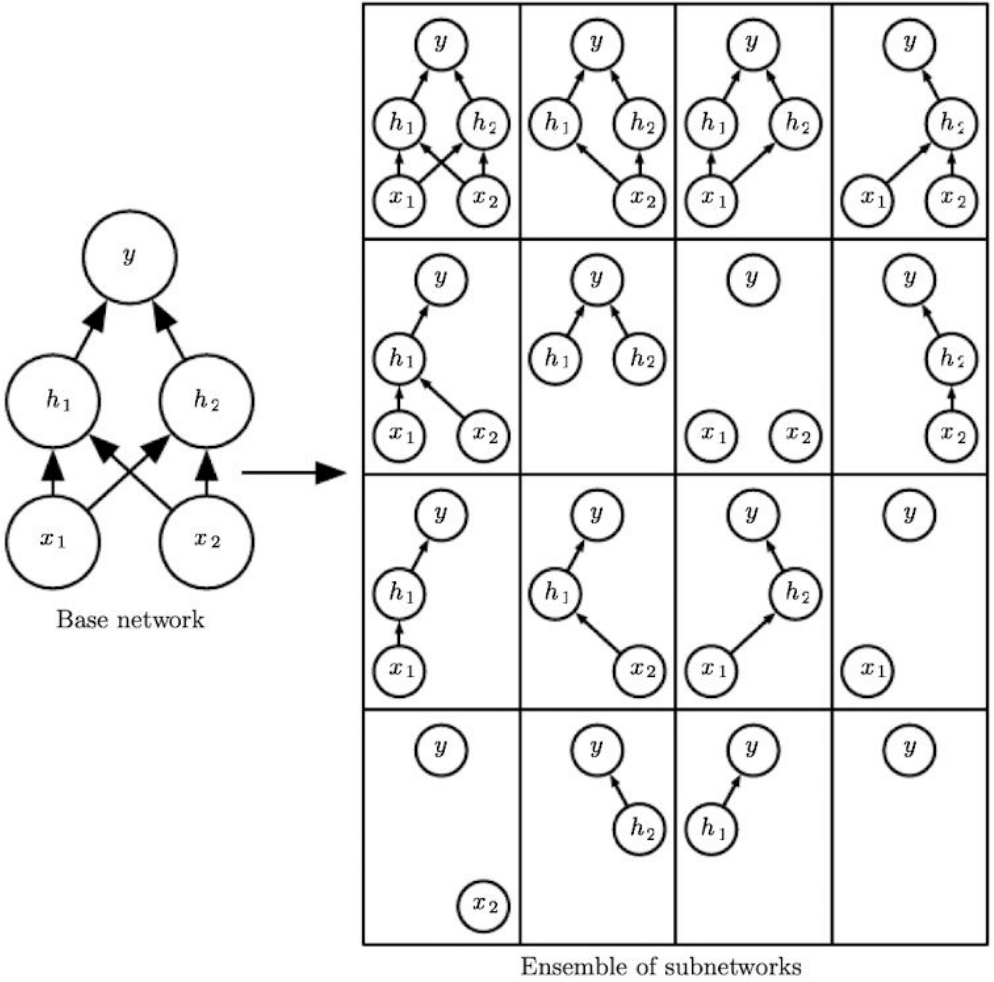


Dropout

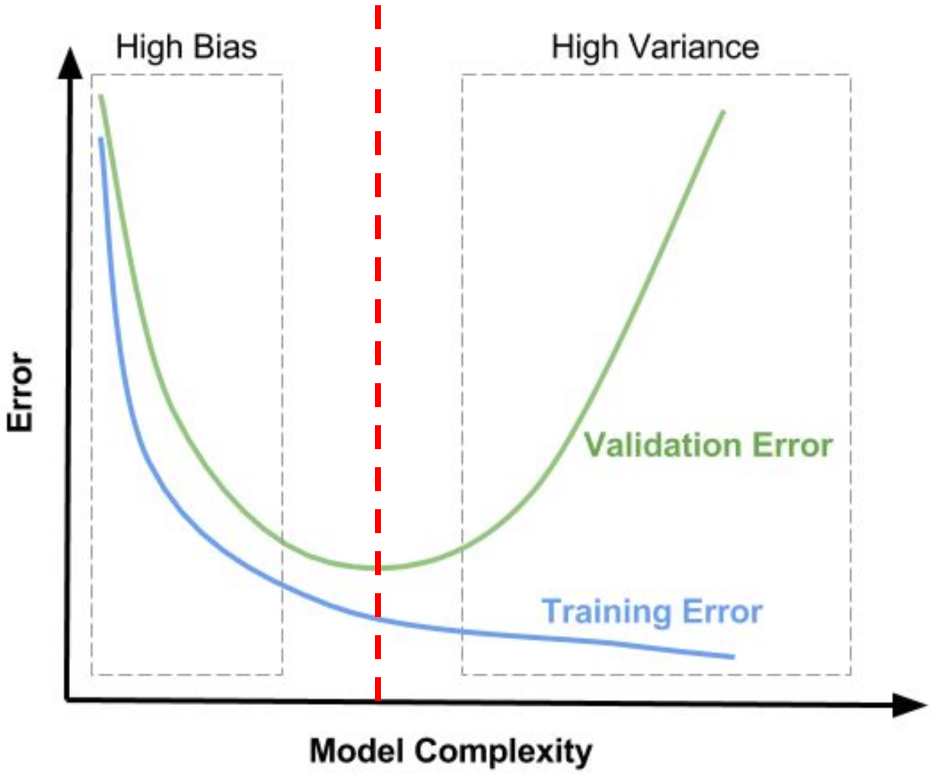


Dropout Only Happens During Training!

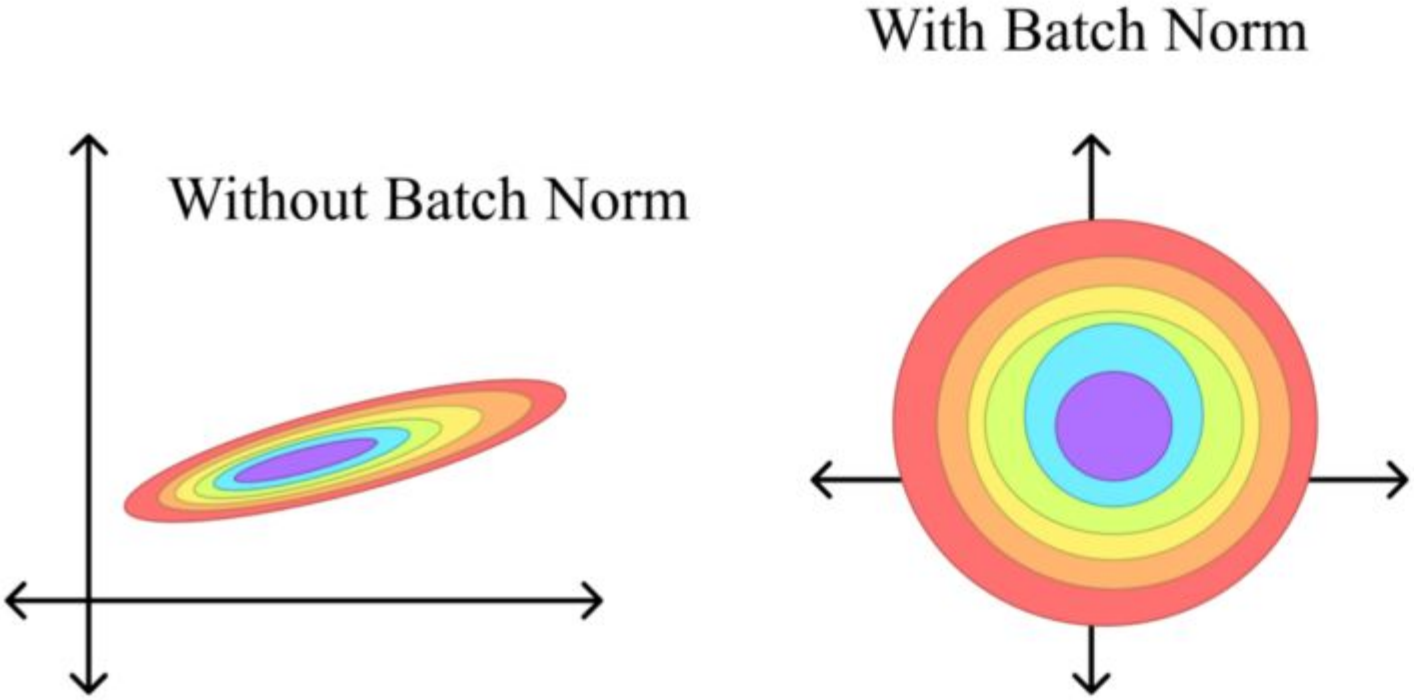
Dropout



Early Stopping



Batch Normalization



From: https://www.linkedin.com/pulse/ways-improve-your-deep-learning-model-batch-adam-albuquerque-lima

Batch Normalization

- Speeds up training
- Lessens the impact of initial weights
- Regularizes your model

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.