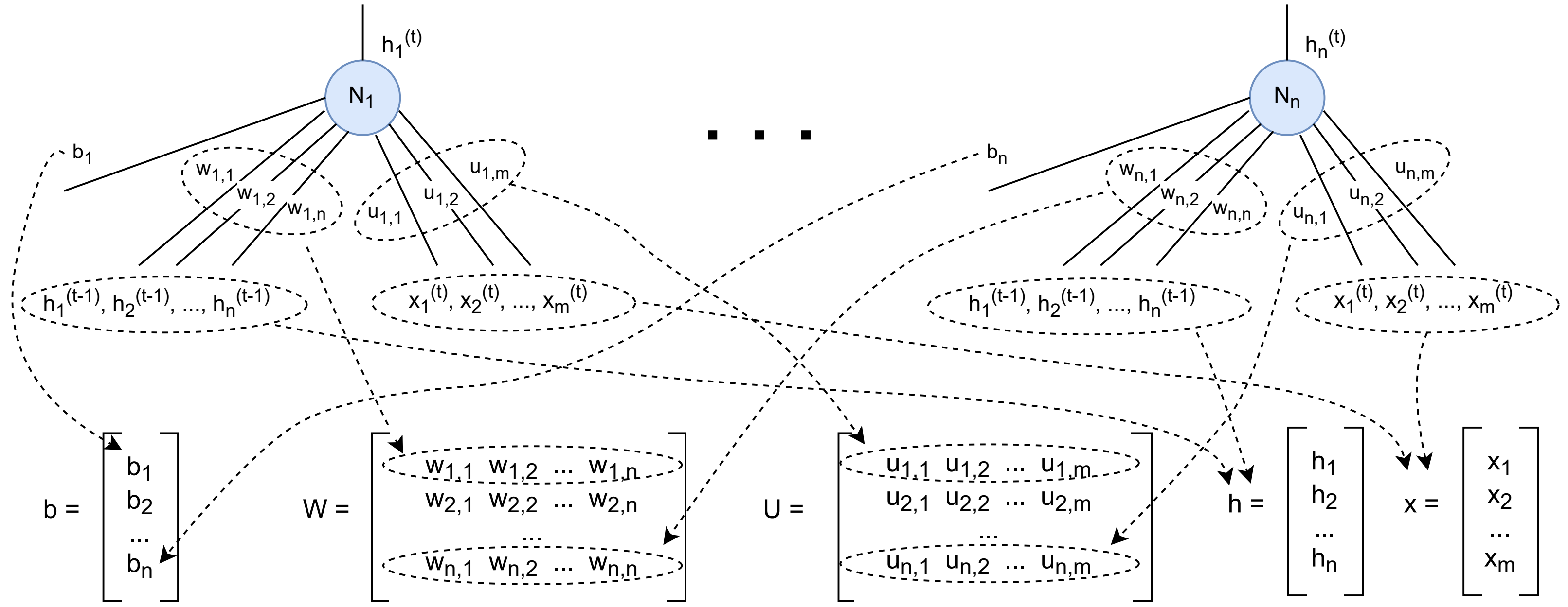
Linear algebra representation

Weighted sum for single neuron: $z = \mathbf{w}\mathbf{x}$

Weighted sums for fully connected layer: $\mathbf{z} = W\mathbf{x}$

Weighted sums for fully connected layer for mini-batch: $Z = WX$

Recurrent layer: $\mathbf{h}^{(t)} = tanh(W\mathbf{h}^{(t-1)} + U\mathbf{x}^{(t)} + \mathbf{b})$



NOTE: Bias term is implicit in all but the recurrent case above

Datasets

Typical splits

Big dataset: 60/20/20 (train/validation/test)

Small dataset: 80/20 (train/test) and k-fold cross-validation

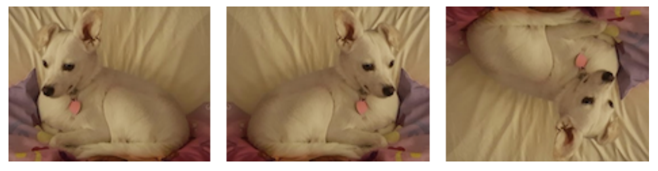
Regularization techniques

L1/L2 Regularization

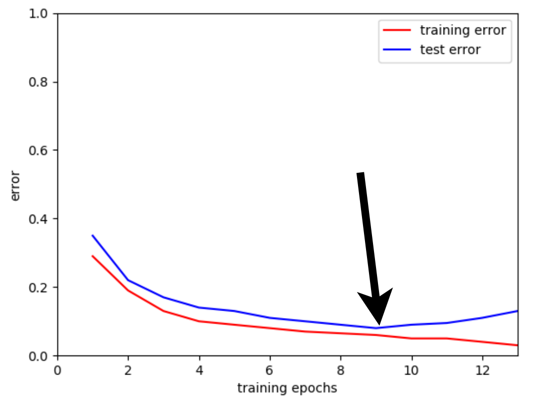
Add a weight penalty to error function:

$L1 = \lambda w \qquad L2 = \lambda w^2$

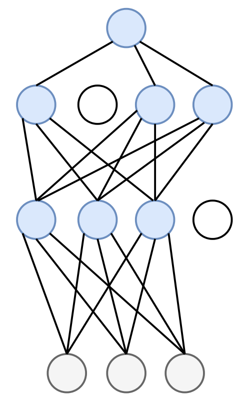
Data augmentation



Early stopping



Dropout



Training algorithm variations

Algorithm	**Description**
Stochastic gradient descent (SGD)	Gradient is computed based on a mini-batch of training examples.
Momentum	Addition to SGD where weight adjustment depends on gradient from previous adjustments as well as the current gradient.
AdaGrad	Variation on SGD that adaptively adjusts the learning rate during training.
Adam	Variation on SGD with both adaptive learning rate and momentum.
RMSProp	Variation on SGD that normalizes gradient using the root mean square (RMS) of recent gradients.

Keeping gradients healthy

Technique	**Mitigates vanishing gradients**	**Mitigates exploding gradients**
Glorot or He weight initialization	Yes	No
Batch normalization	Yes	No
Nonsaturating neurons, e.g., ReLU	Yes	No
Gradient clipping	No	Yes
Constant error carousel	Yes	Yes
Skip connections	Yes	No