LINEAR ALGEBRA FOR AI/ML

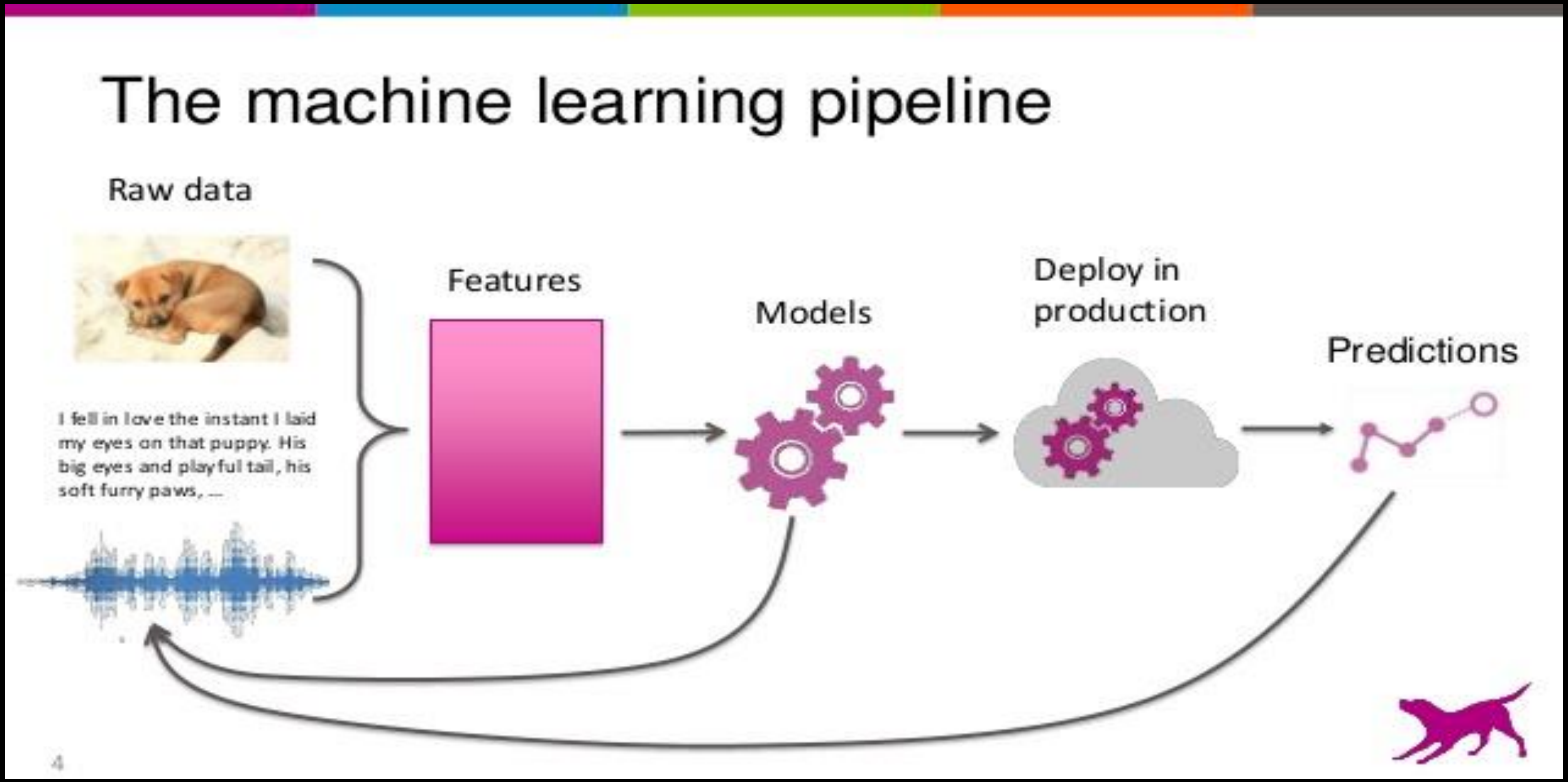
Jiaul Paik

TOPICS TODAY

- **Basics of Eigenvalue & Eigenvector**
- **Data reduction**
  - **Auto-encoder**
  - **Principal Component Analysis**

Autoencoder using Deep Neural Network

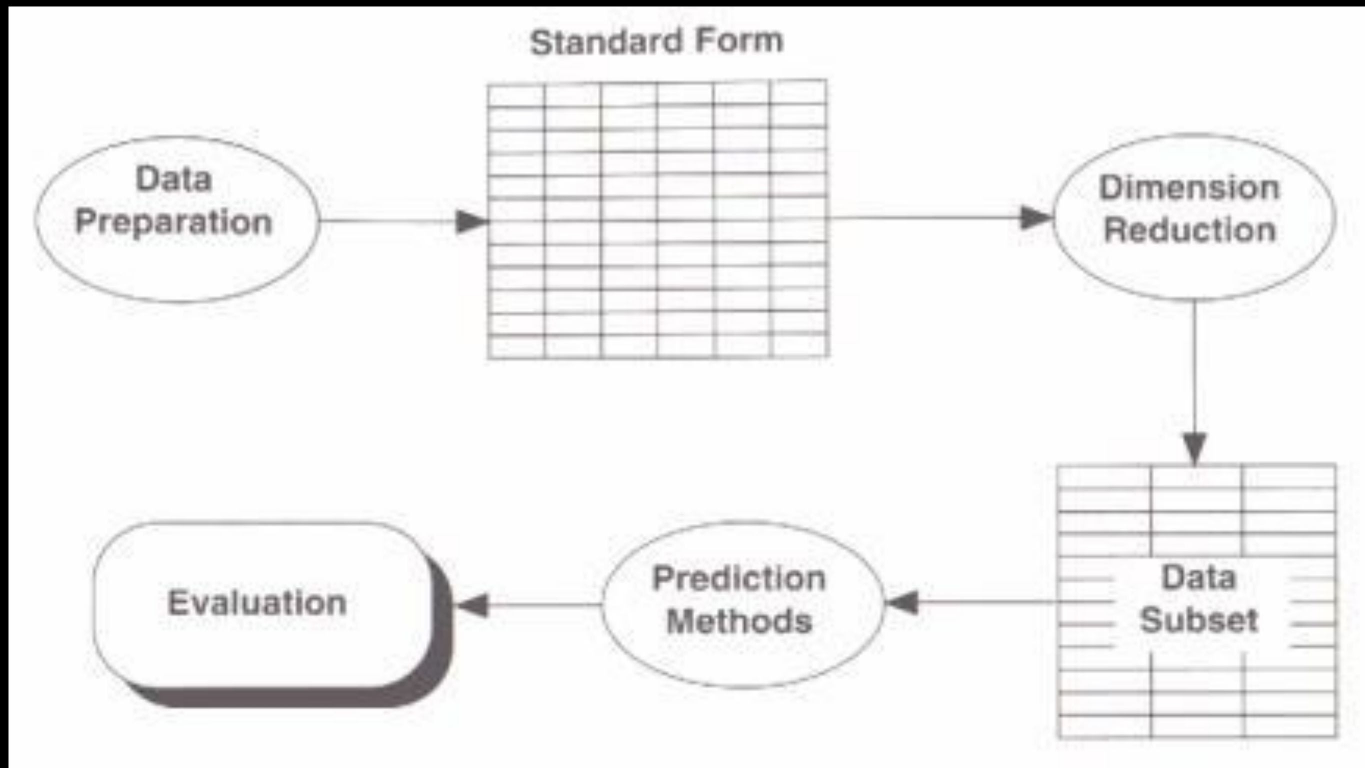
Feature Engineering: Feature Extraction



Features are generated by applying some function on the raw data

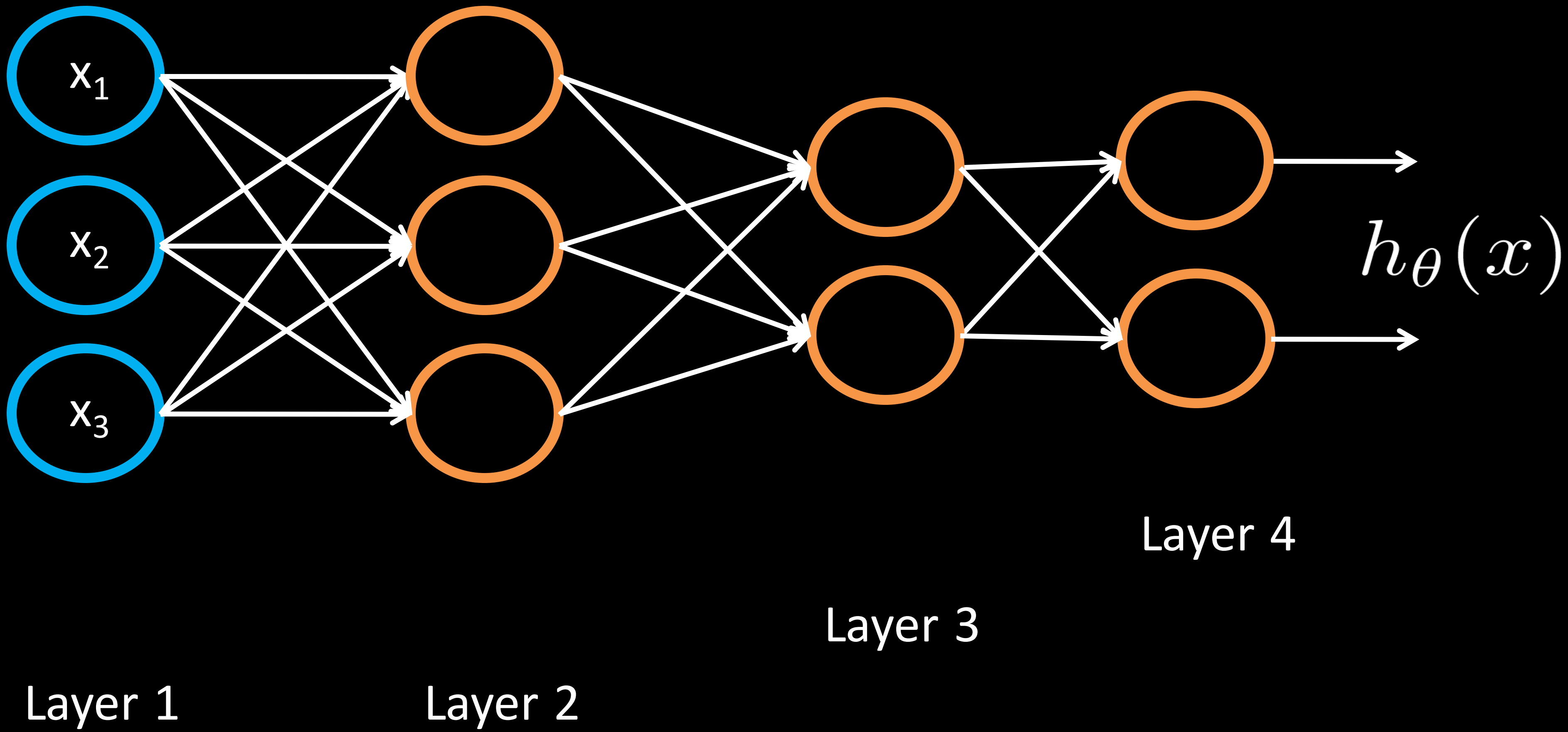
Image Example: Average of 3×3 sub matrices

Feature/Dimensionality Reduction

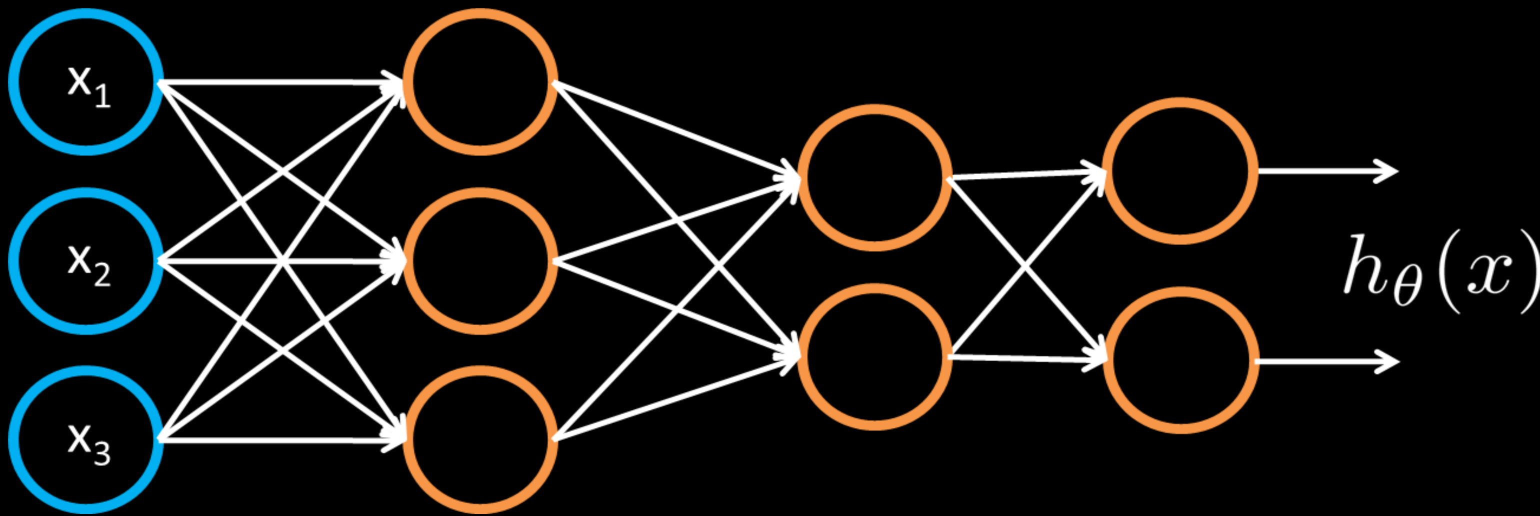


Neural Network

Example 4 layer network with 2 output units:



Training a neural network



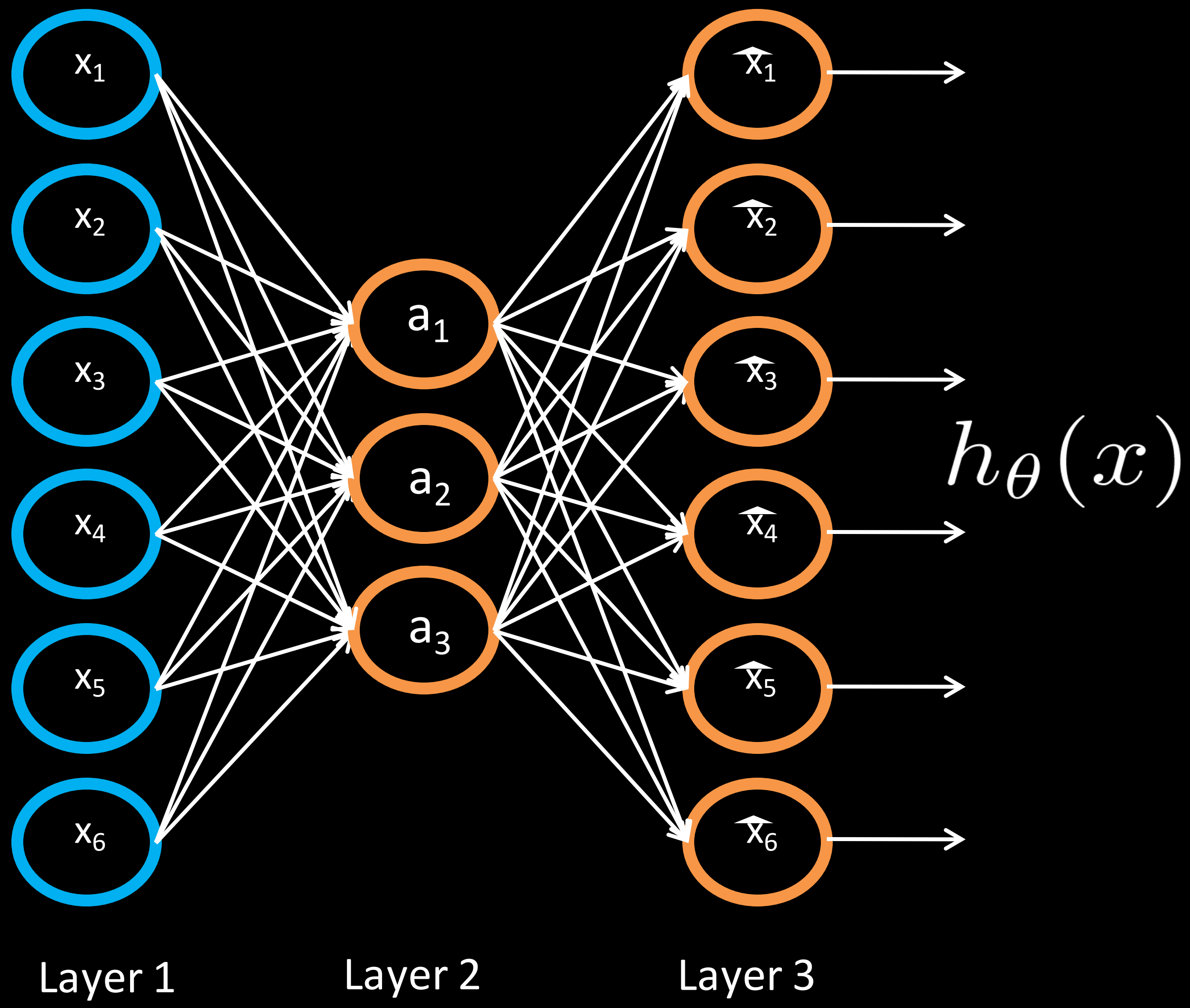
Given training set (x_1, y_1), (x_2, y_2), (x_3, y_3), ….

Adjust parameters θ (for every node) to make:

$$h_\theta(x_i) \approx y_i$$

(Use gradient descent. “Backpropagation” algorithm.)

Feature generation/reduction with a neural network



Autoencoder.

Network is trained to output the input (learn identify function).

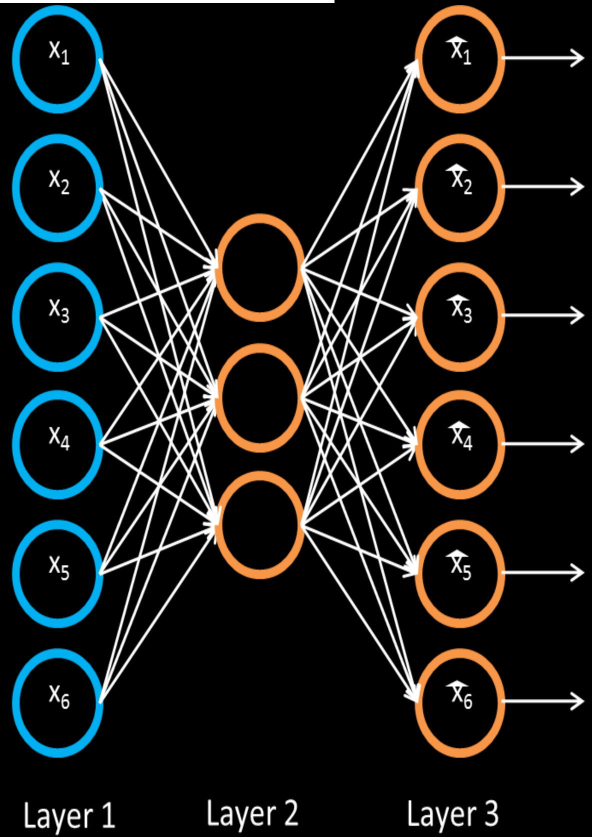
$$h_\theta(x) \approx x$$

Feature generation/reduction with a neural network

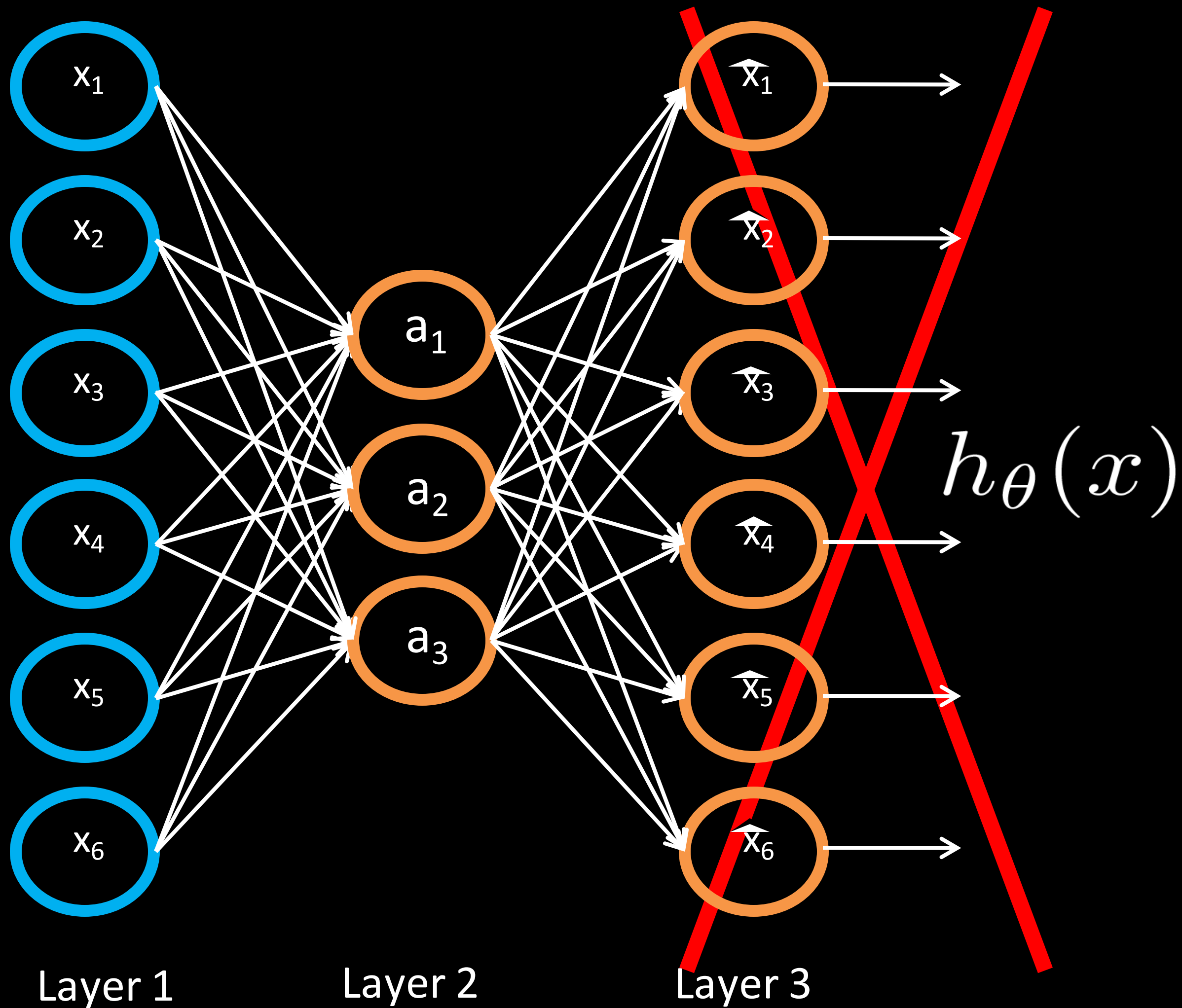
Training a sparse autoencoder.

Given unlabeled training set x_1, x_2, …

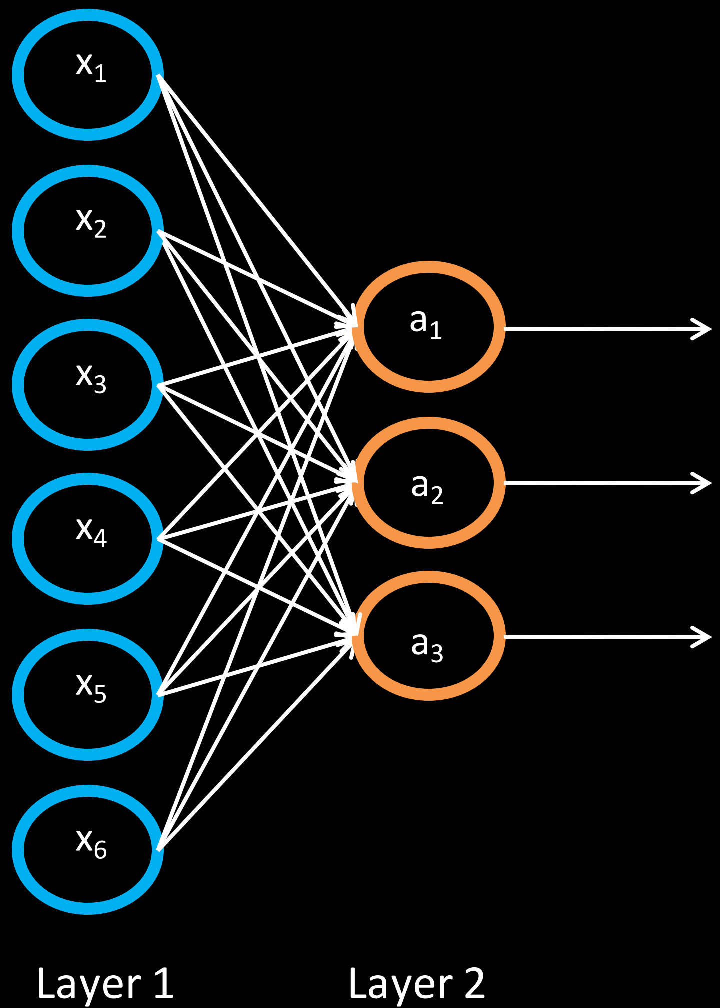
Objective function: $\min_\theta \underbrace{\|h_\theta(x) - x\|^2}_{\text{Reconstruction error term}}$



Feature generation/reduction with a neural network



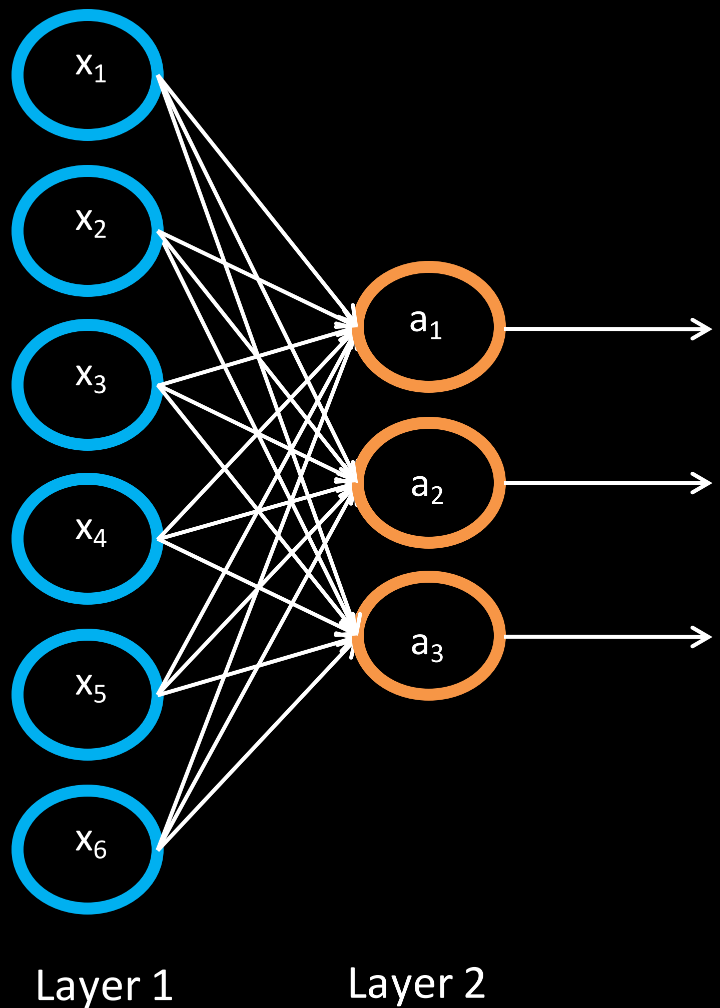
Feature generation/reduction with a neural network



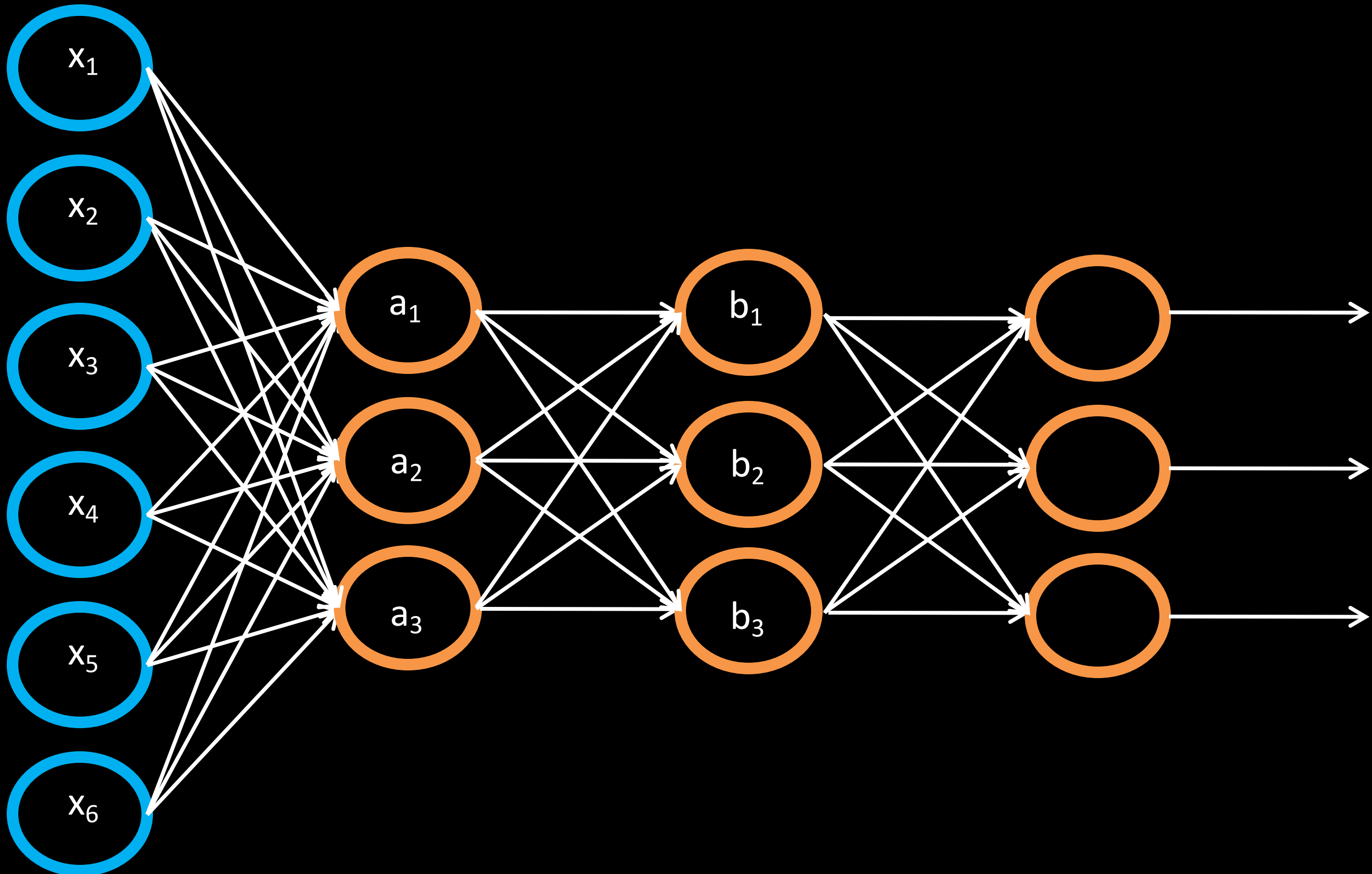
$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

New representation for input.

Feature generation/reduction with a neural network



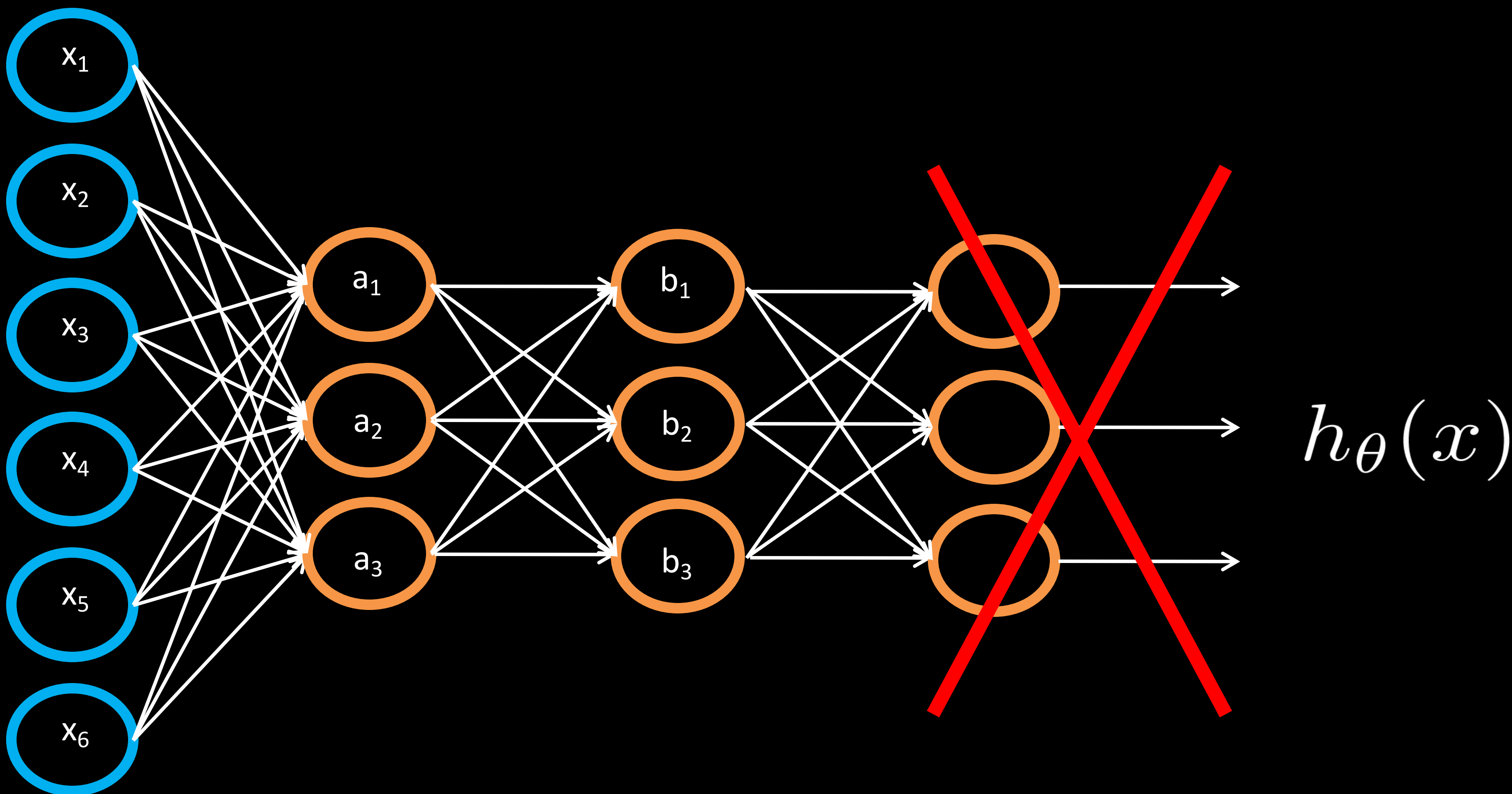
Feature generation/reduction with a neural network



$h_\theta(x)$

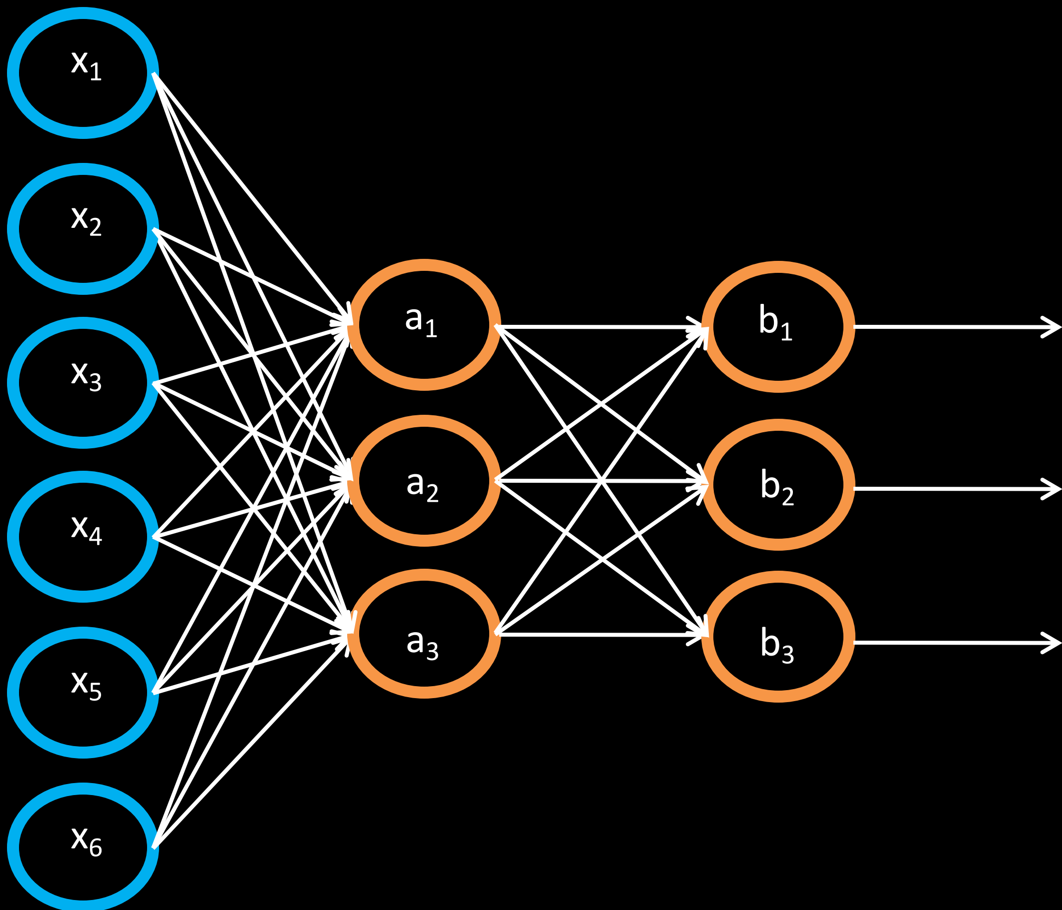
Train parameters so that $h_\theta(x) \approx a$

Feature generation/reduction with a neural network



Train parameters so that $h_\theta(x) \approx a$

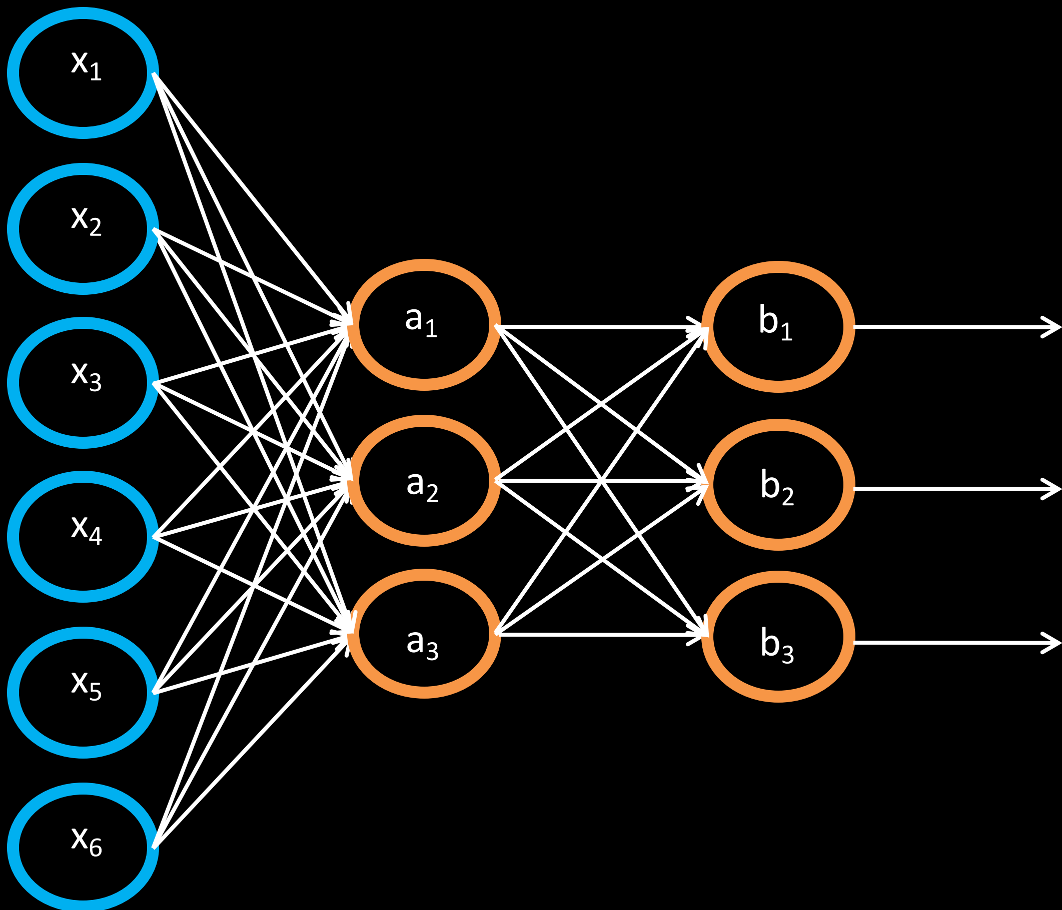
Feature generation/reduction with a neural network



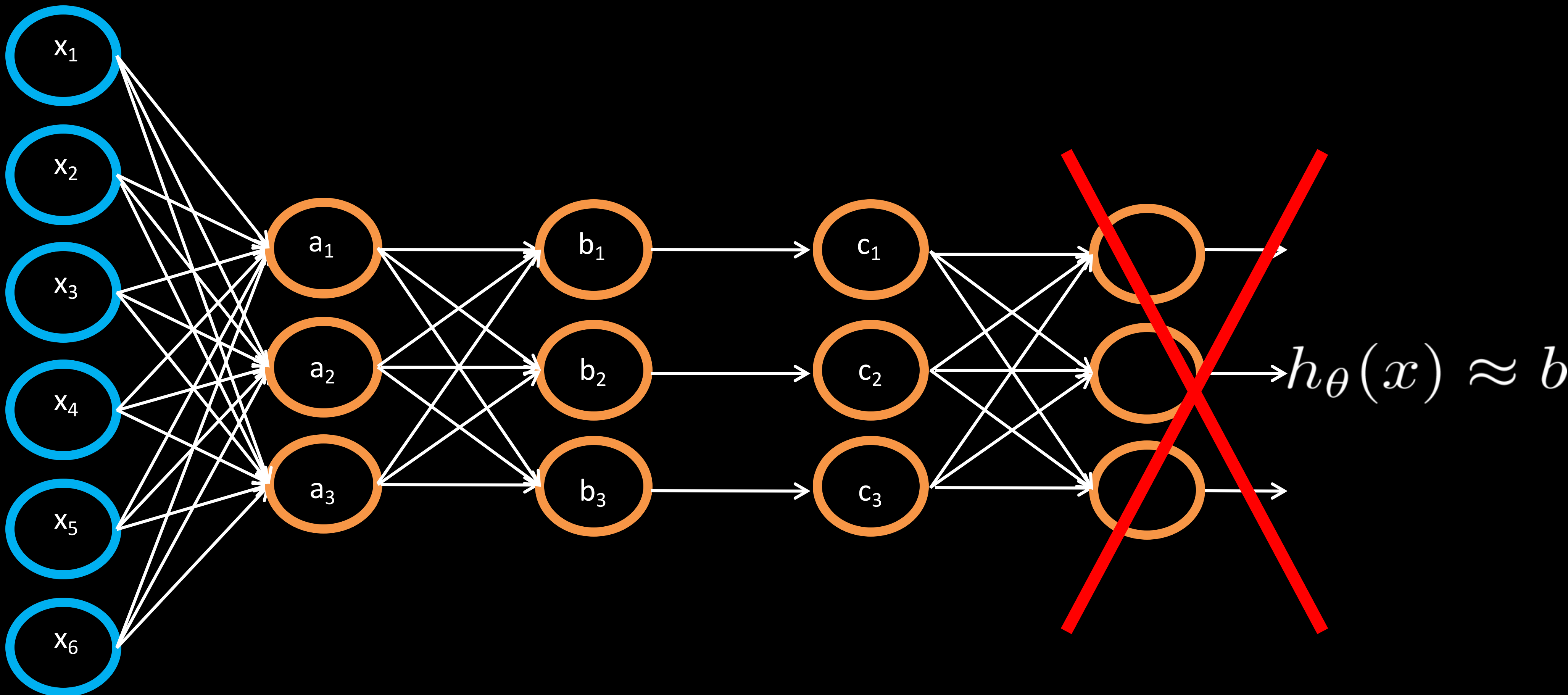
$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

New representation for input.

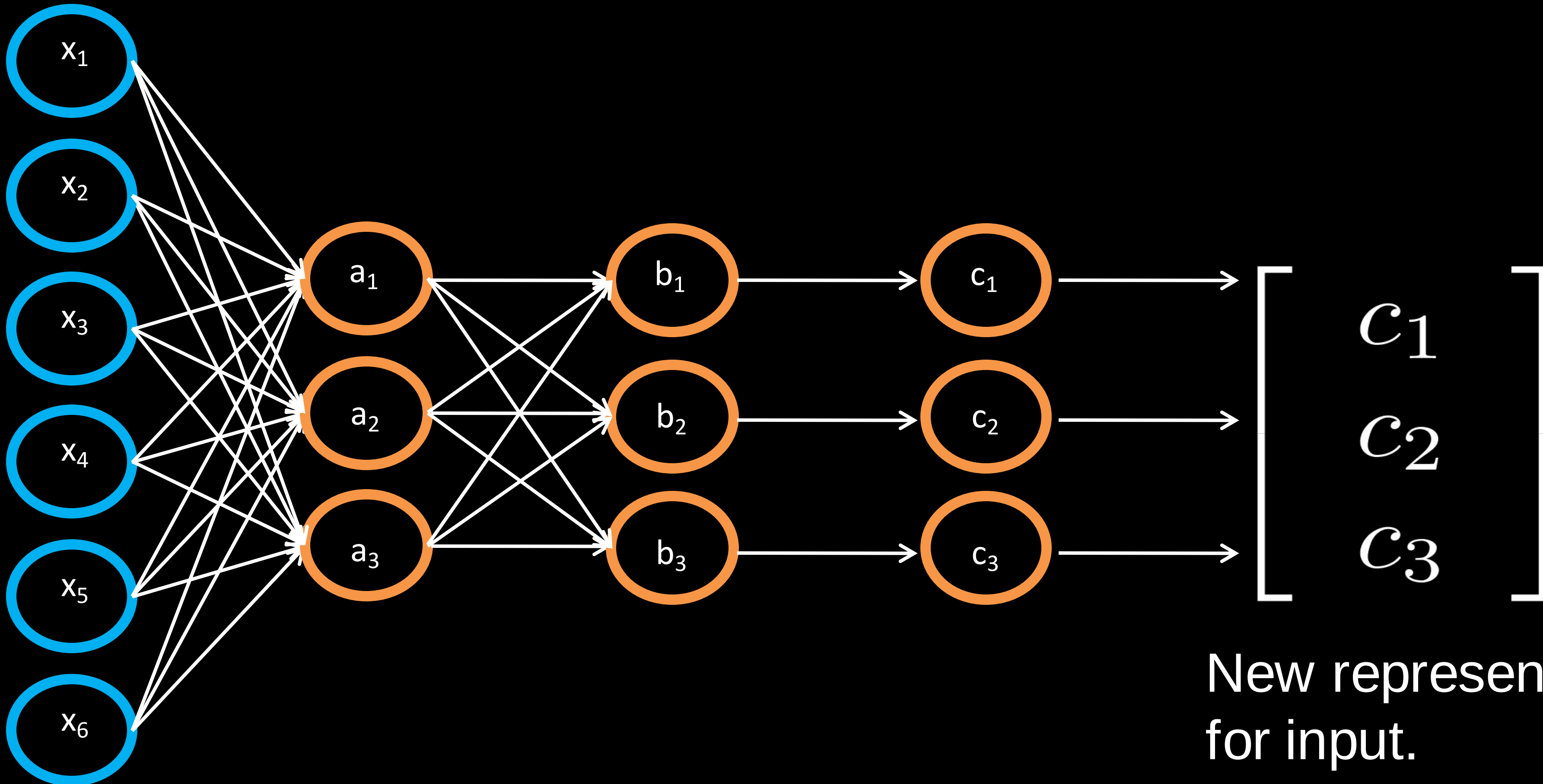
Feature generation/reduction with a neural network



Feature generation/reduction with a neural network



Feature generation/reduction with a neural network



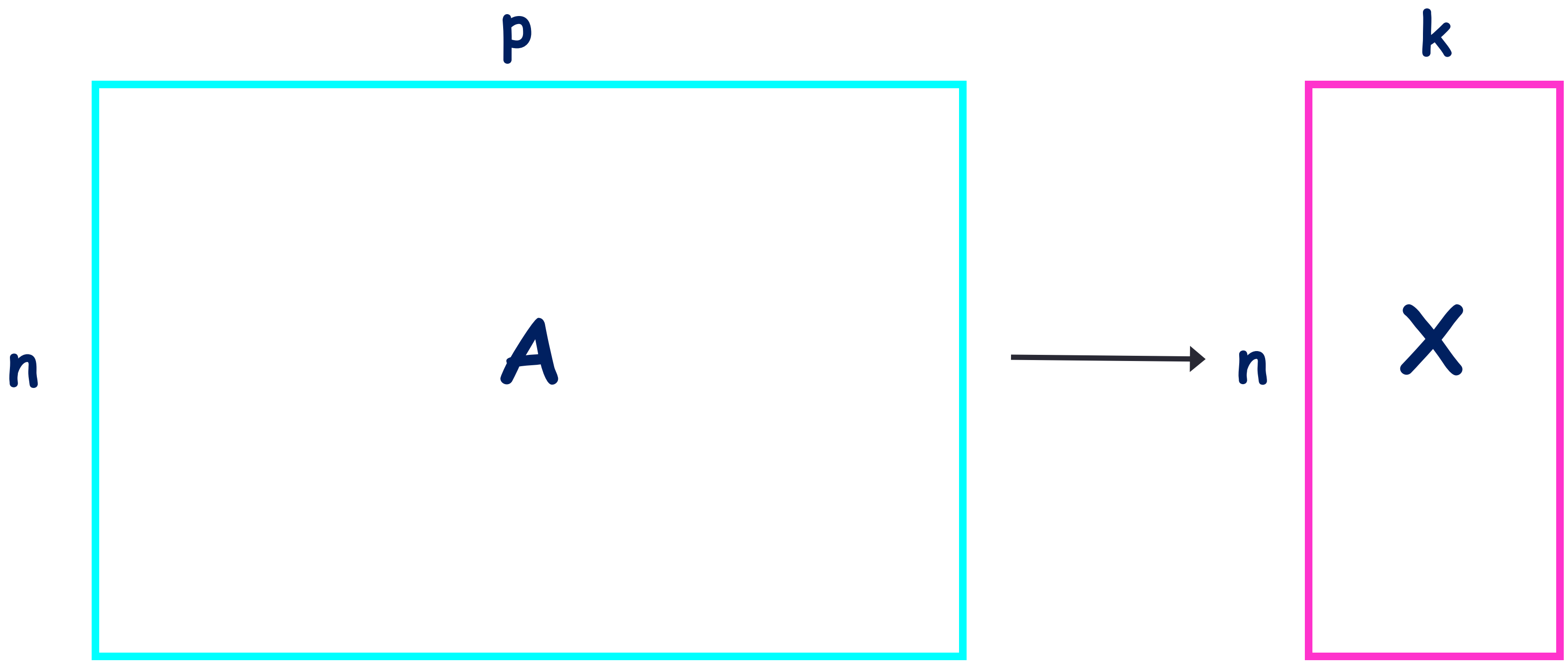
New representation for input.

Use [c_1, c_3, c_3] as representation to feed to learning algorithm.

Principal Component Analysis (PCA)

Data Reduction

- **Summarization of data with many (p) variables by a smaller set of (k) derived (latent, composite) variables.**

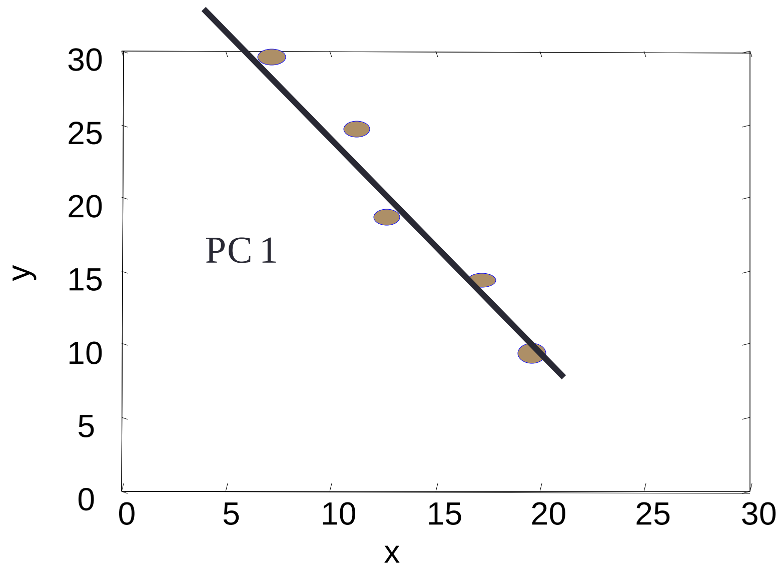


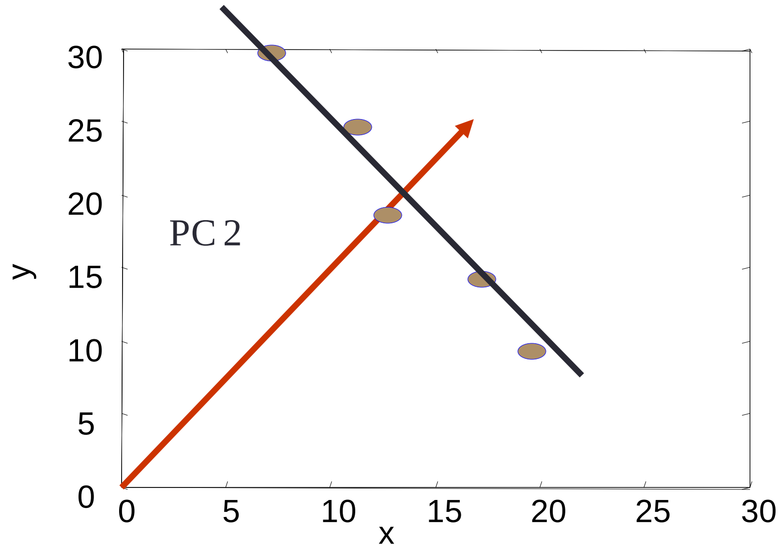
Data Presentation: Key questions?

- **Do we need a all n-dimension space to view data?**
- **Better presentation (new axes) than original axes?**
- **How to find the 'best' low dimension space that conveys maximum useful information?**
- **One answer: Find "Principal Components"**

Principal Components

- All principal components (PCs) start at the origin
- First PC is direction of maximum variance
- Subsequent PCs are orthogonal to 1st PC and describe maximum residual variance





The Goal

We wish to summarize the underlying variance-covariance structure of a large set of variables through a few linear combinations of these variables.

Trick: Rotate Coordinate Axes

- Suppose we have p features $x_1,\ldots,x_p$.
- Our goal is to develop a new set of p axes (linear combinations of the original p axes) in the directions of greatest variability:
- This is accomplished by rotating the axes.