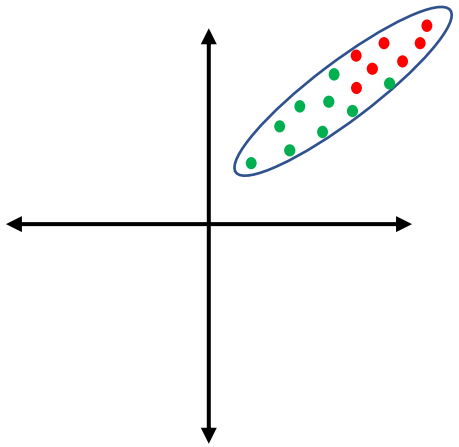
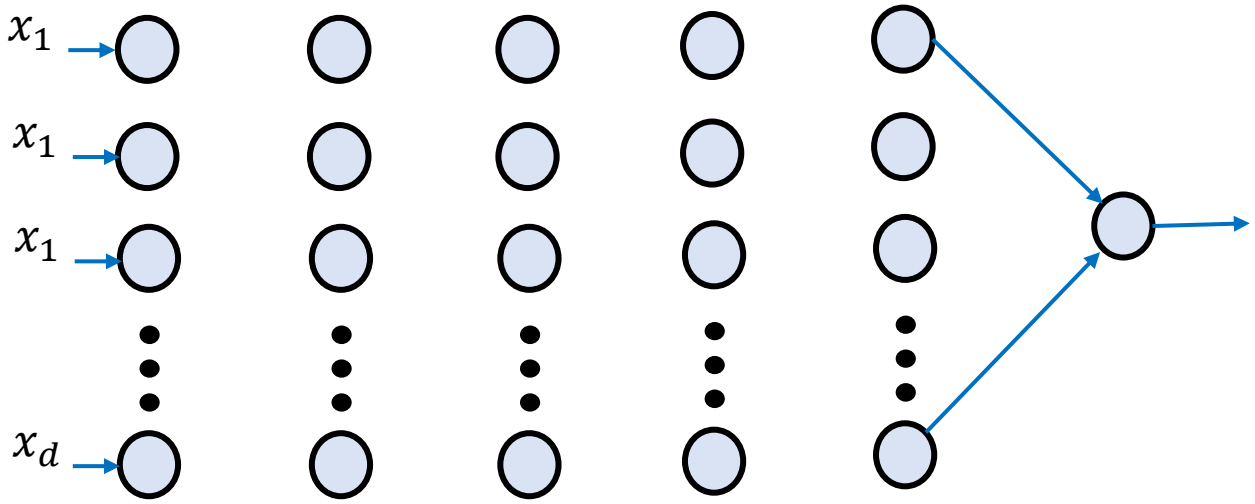
# Batch Normalization



'd' dimensional feature space drawn in 2 dimension




Deep classifier network

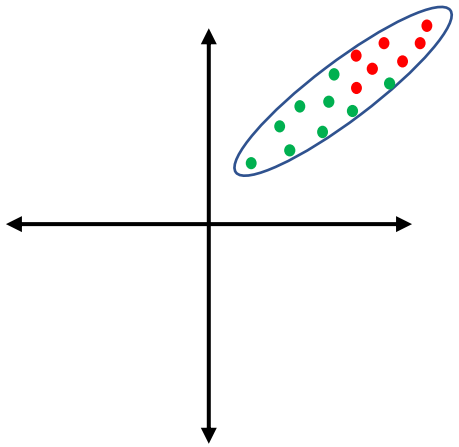
The range of different input features can be very different. Consider the below example. The feature distribution in this case looks like the top left plot. Please note that we are plotting only 2-dimensional distribution while in reality, it can be 'd' dimensional.

**Example:** Classify whether a person get a chance in school basketball team

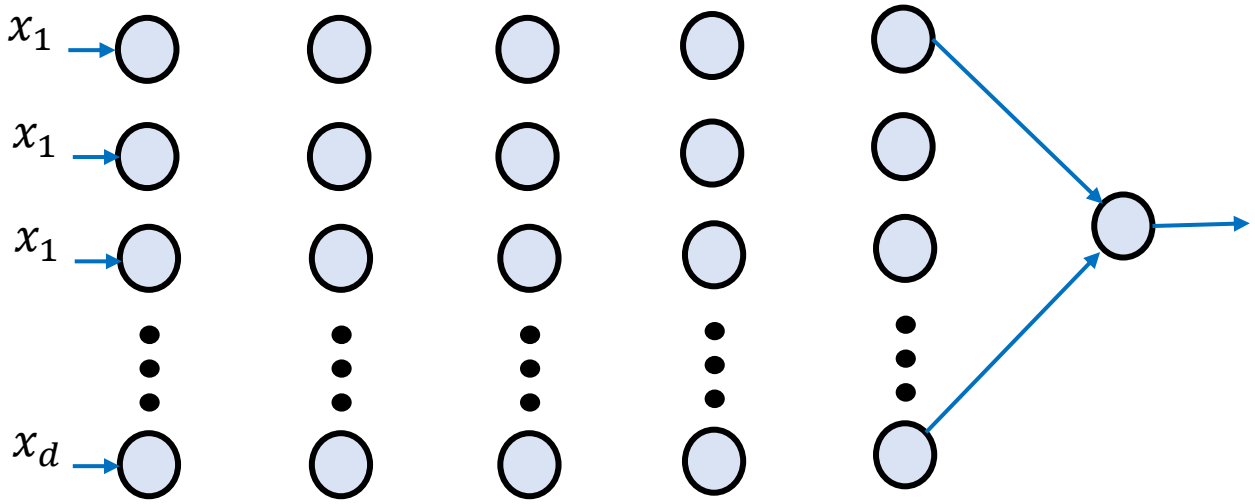
**Features:**

- Age: range 0-100 years
- Height: range 0-8 feet
- Weight: range 0 – 300 lb

# Batch Normalization



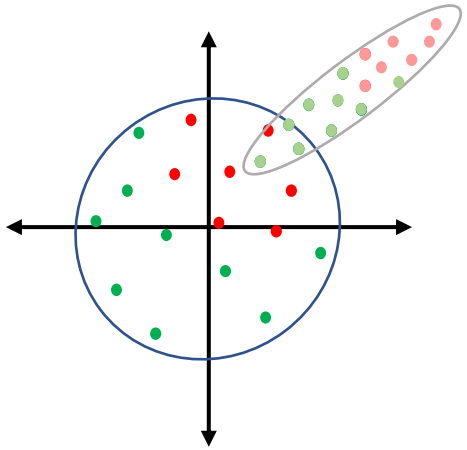
'd' dimensional feature space drawn in 2 dimension



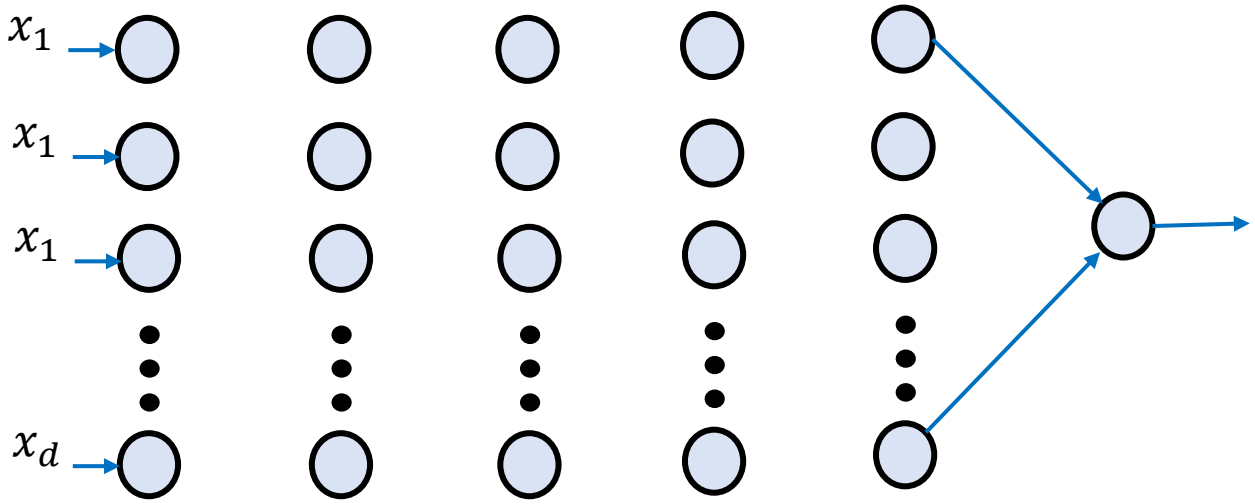

Deep classifier network

Different feature ranges can **translate** the sample distribution from the origin and distort the shape of the distribution. This has a lot of adverse effects during network optimization. Finding a decision boundary becomes harder in this case and optimization takes a lot of time.

# Batch Normalization



'd' dimensional feature space drawn in 2 dimension




Deep classifier network

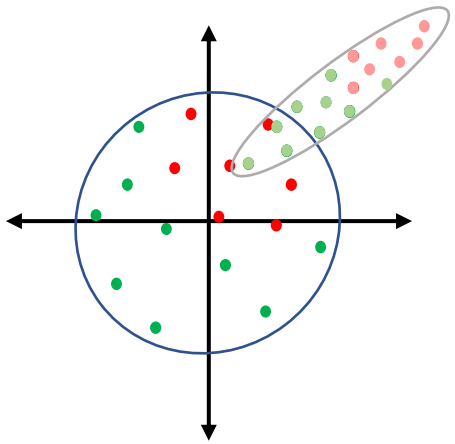
**Solution:** perform normalization of input data before passing it to the network. The specific normalization method shown below is called zero mean unit standard deviation. As a result of the normalization, the input distribution looks like the top left figure. Here $\boldsymbol{\mu}$ & $\boldsymbol{\sigma}$ are the mean and standard deviation of the training samples.

Zero mean, unit standard deviation

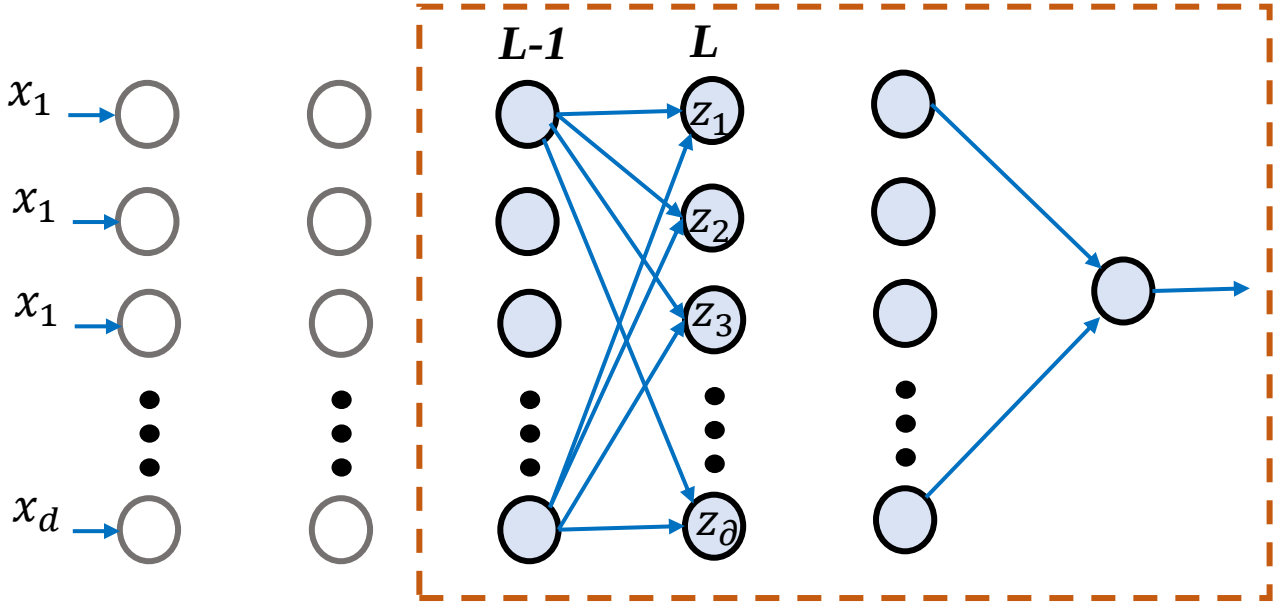
$$\overset{N}{\tilde{\boldsymbol{x}}^i} = \frac{\boldsymbol{x}^i - \boldsymbol{\mu}}{\boldsymbol{\sigma}}$$

$\overset{N}{\tilde{\boldsymbol{x}}^i}$ :Normalized input

# Batch Normalization



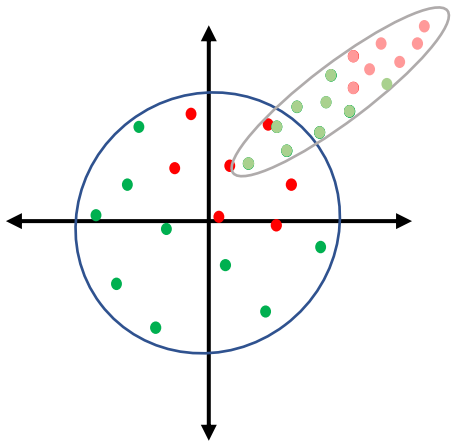
'd' dimensional feature space drawn in 2 dimension



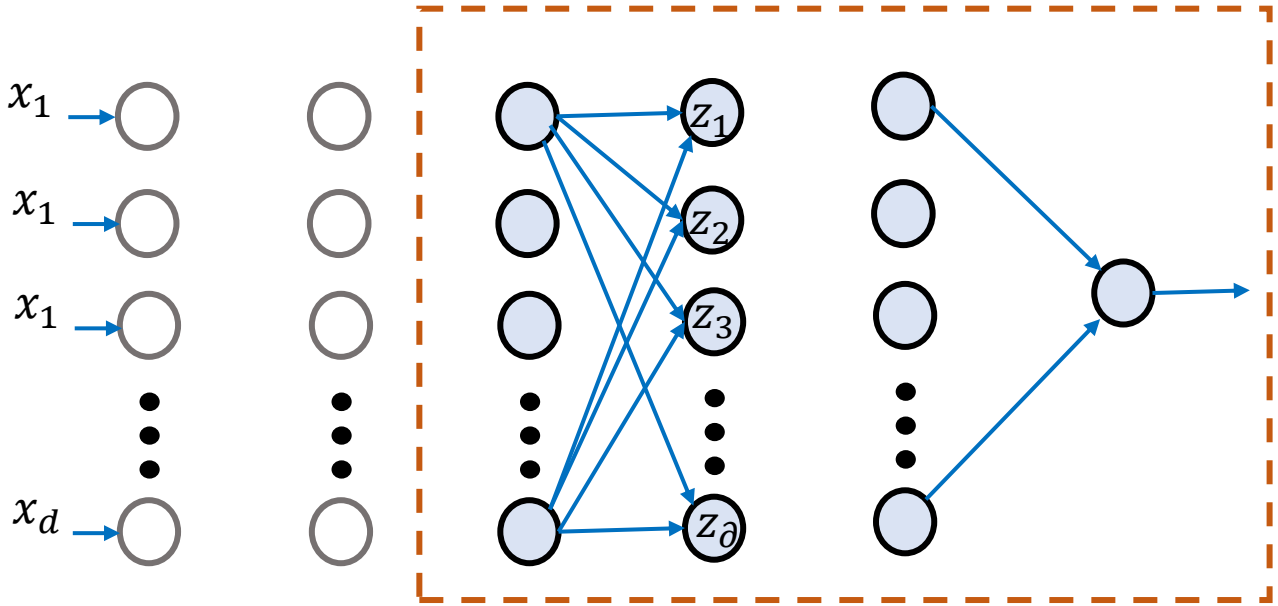

Consider only part of the network in the dotted box

However, normalizing the input data doesn't solve the full problem. The above figure explains the problem. Consider layer $\boldsymbol{L}$ of the network. During the forward pass, this layer always receives inputs ($\boldsymbol{z} = [z_1, z_2, \ldots, z_\partial]$) from the previous layer. However, the input of layer $\boldsymbol{L}$ is the output of layer $\boldsymbol{L}$**-1** multiplied by connection weights. During the training phase, connection weights of layers **1 to** $\boldsymbol{L}$**-1** change after every batch of training. This causes the change of input distribution of layer $\boldsymbol{L}$. This is called **covariate shift** which is observed in every hidden layer of a deep neural network.

# Batch Normalization



'd' dimensional feature space drawn in 2 dimension




Consider only part of the network in the dotted box

**Solution:** normalization of input distribution for every hidden layer. Since, because of the computational cost, this normalization cannot be performed on all training samples, it is performed on each batch of the training sample. The batch normalization formula is shown below.

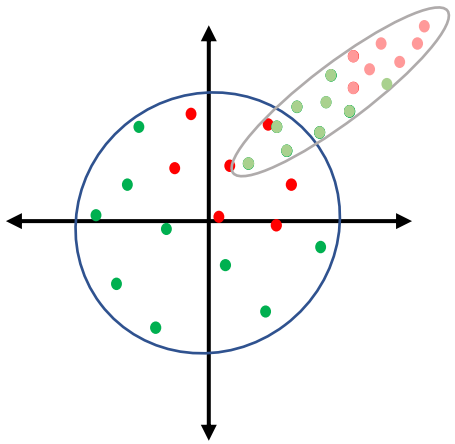
Zero mean, unit standard deviation for hidden layer input

$$\overset{N}{\tilde{\boldsymbol{z}}^i} = \frac{\boldsymbol{z}^i - \boldsymbol{\mu}_{batch}}{\boldsymbol{\sigma}_{batch}}$$

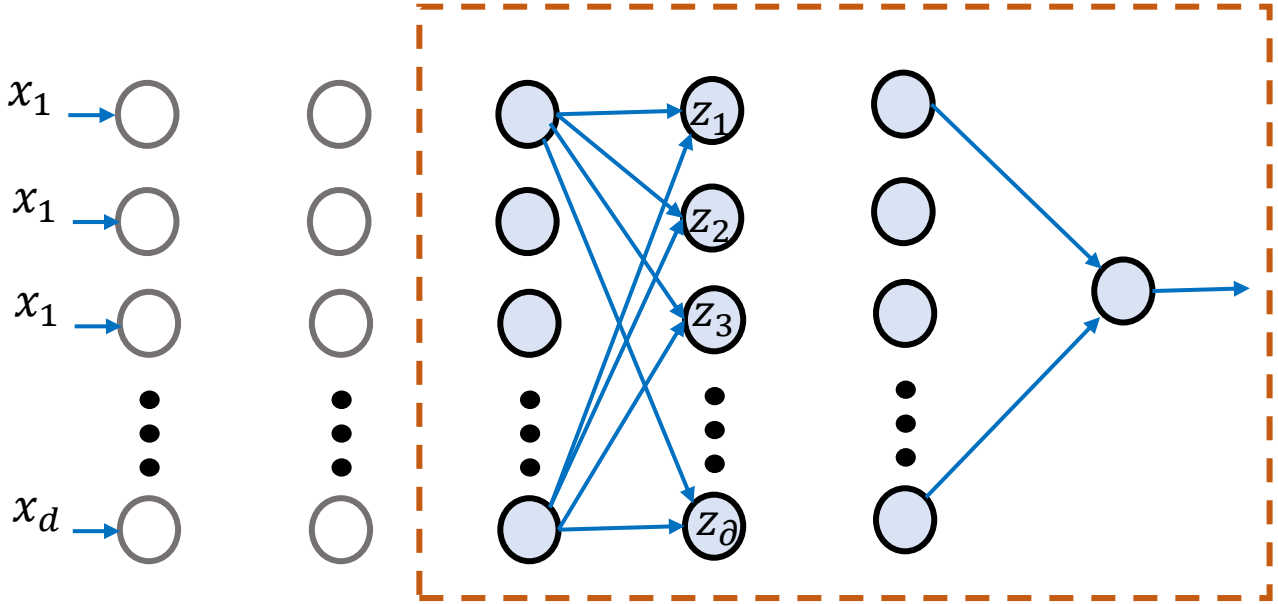
$\boldsymbol{\mu}_{batch}$: mean of a batch of inputs at layer $\boldsymbol{L}$

$\boldsymbol{\sigma}_{batch}$: standard deviation of a batch of inputs at layer $\boldsymbol{L}$

# Batch Normalization



'd' dimensional feature space drawn in 2 dimension




Consider only part of the network in the dotted box

The batch normalization solution shown in the previous slide is incomplete. Sometimes network may want to learn some distribution that is not zero mean and unit standard deviation. To allow the network to have that flexibility, two new learnable parameters $\gamma$ & $\beta$ are introduced in the batch normalization step. The final formulation is shown below.

$$\overset{N}{\tilde{\boldsymbol{z}}^i} = \left(\frac{\boldsymbol{z}^i - \boldsymbol{\mu}_{batch}}{\boldsymbol{\sigma}_{batch}}\right).\gamma + \beta$$

# Batch Normalization

**Advantage of batch normalization:**

- Faster learning rate
- Reduced overfitting because of regularization effect

**Standard practice for CNN:**

- Use batch normalization in convolutional layers
- Use dropout regularization in dense layers