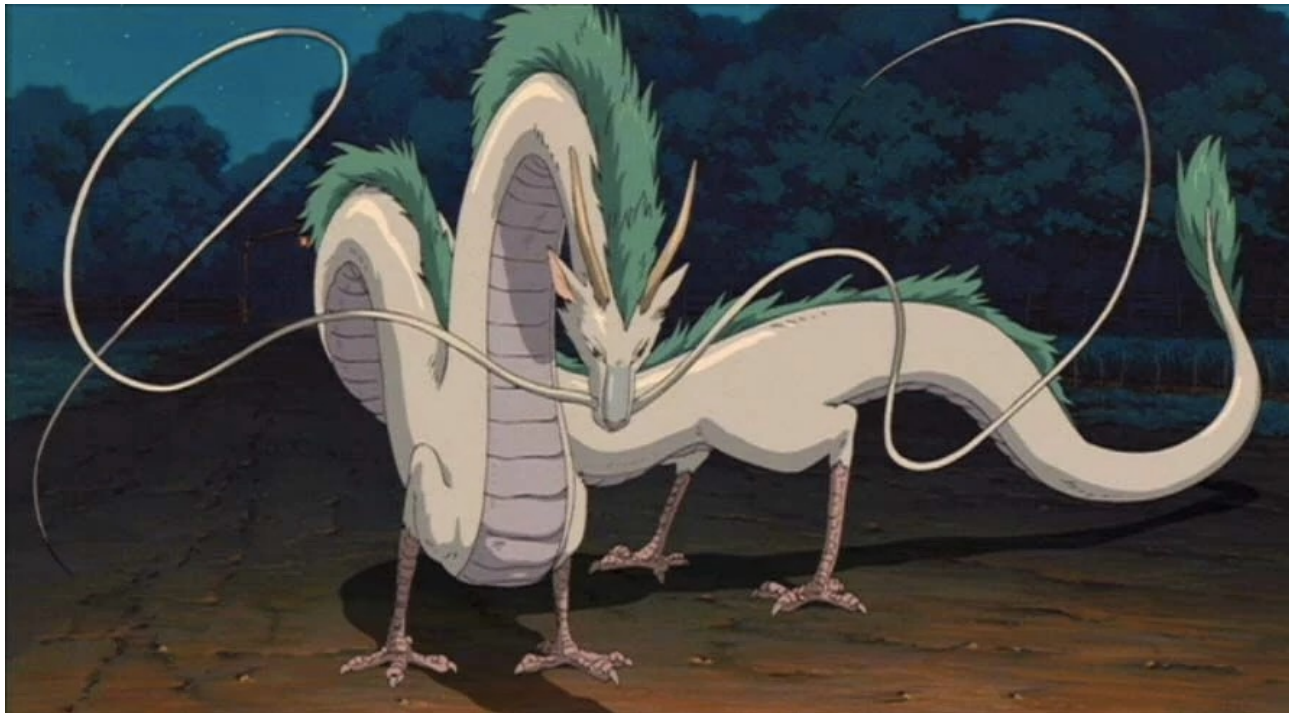
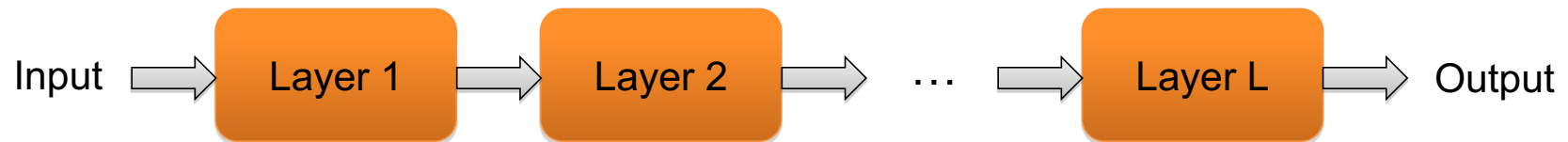


Multi-layer networks and hyperparameter search

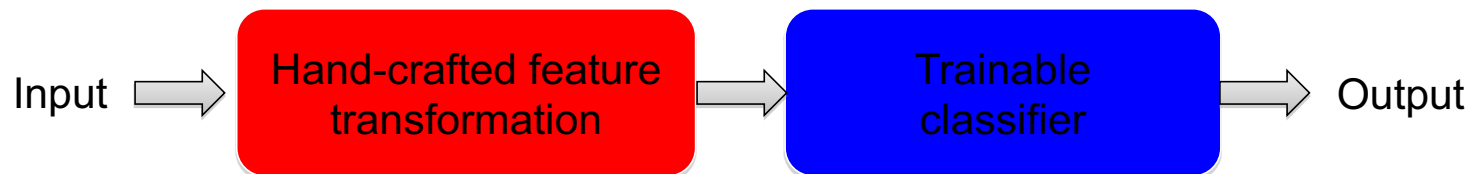


From linear to nonlinear classifiers

- To achieve good accuracy on challenging problems, we need *nonlinear* models
 - Deep learning philosophy: compose simple feature transformation layers into a nonlinear function that is *end-to-end differentiable*



- Contrast with historical “shallow learning” philosophy: transform features via a hand-crafted non-trainable function and then apply simple trainable classifier on top

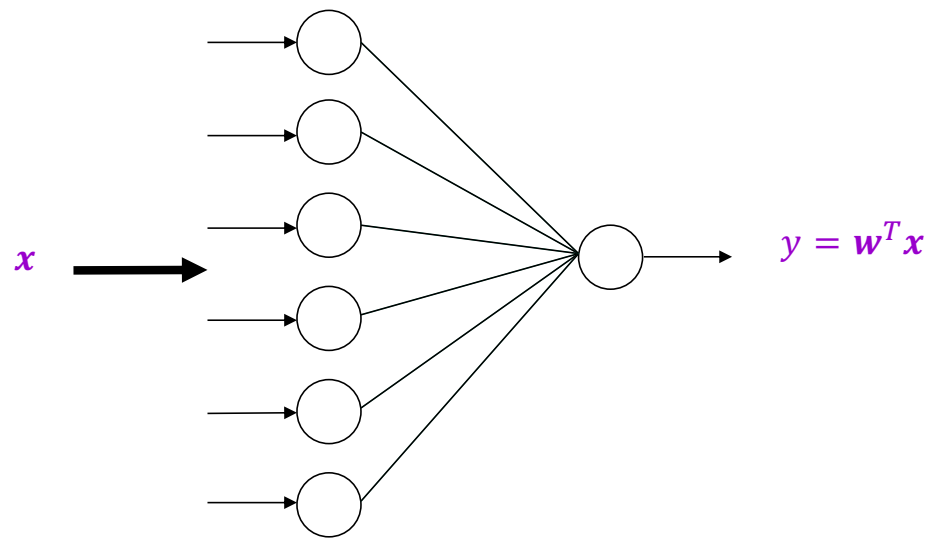


Overview

- Multi-layer networks
 - Linear layers
 - Activation functions
- Hyperparameter search, generalization

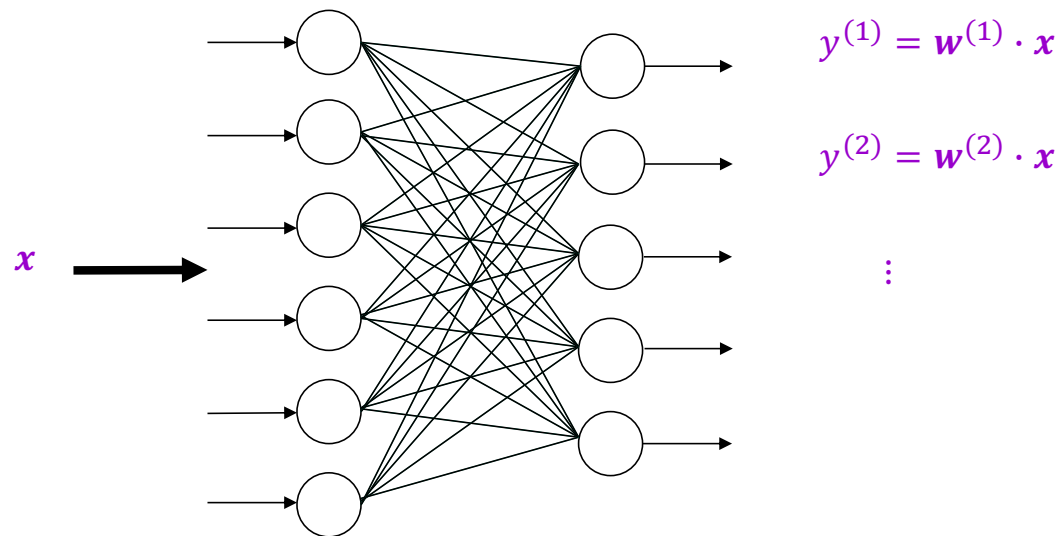
From linear units to multi-layer networks

Linear unit

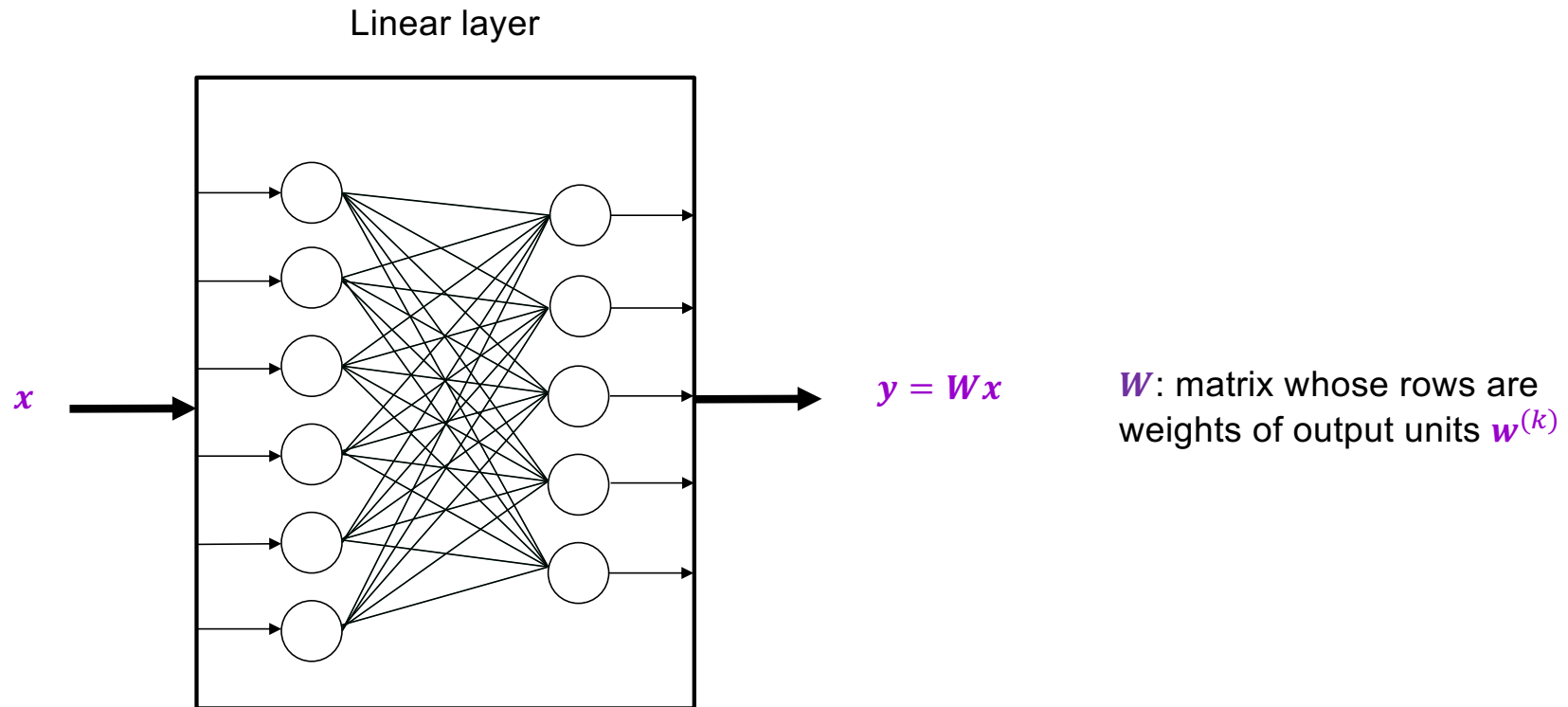


From linear units to multi-layer networks

Bank of linear units

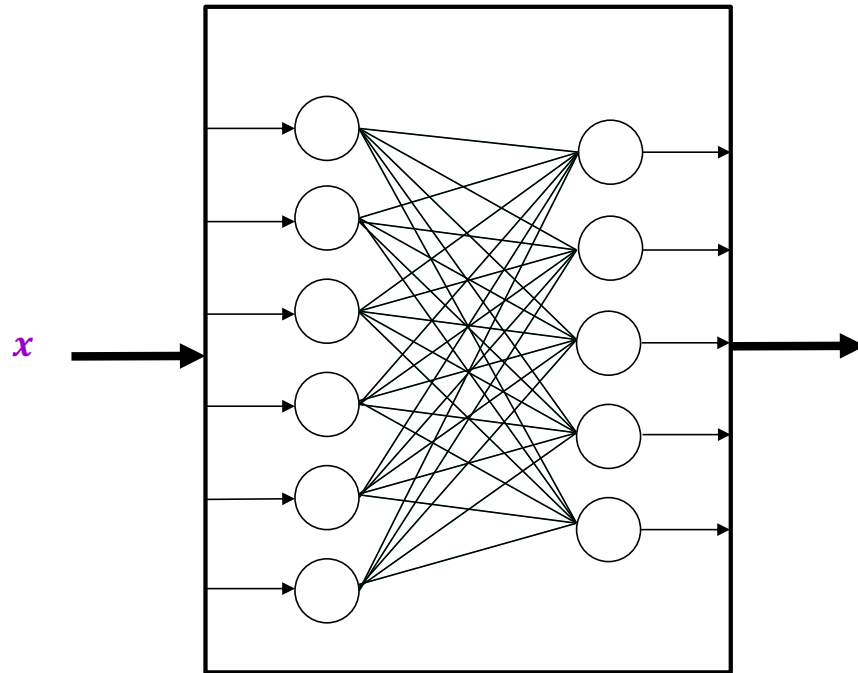


From linear units to multi-layer networks



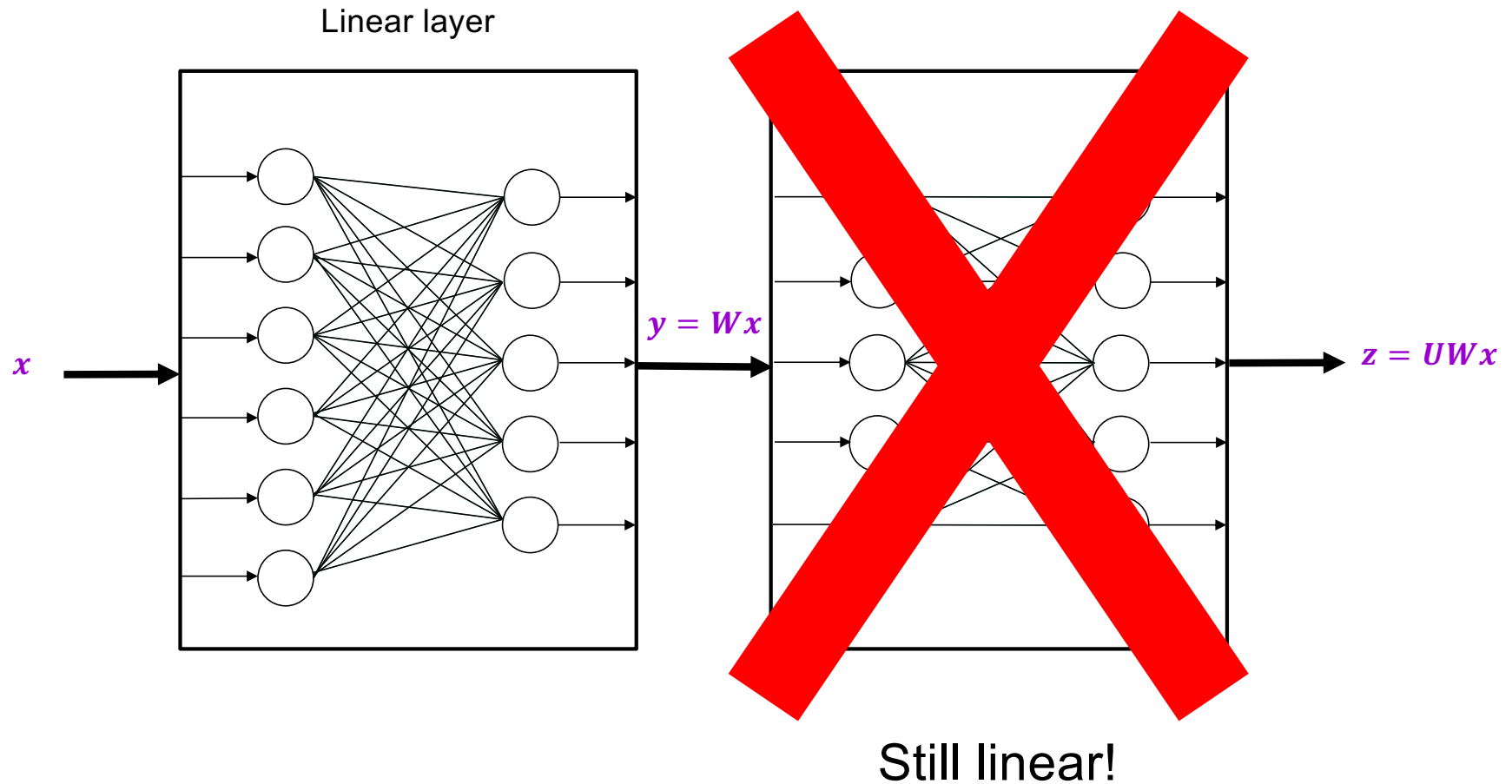
From linear units to multi-layer networks

Linear layer

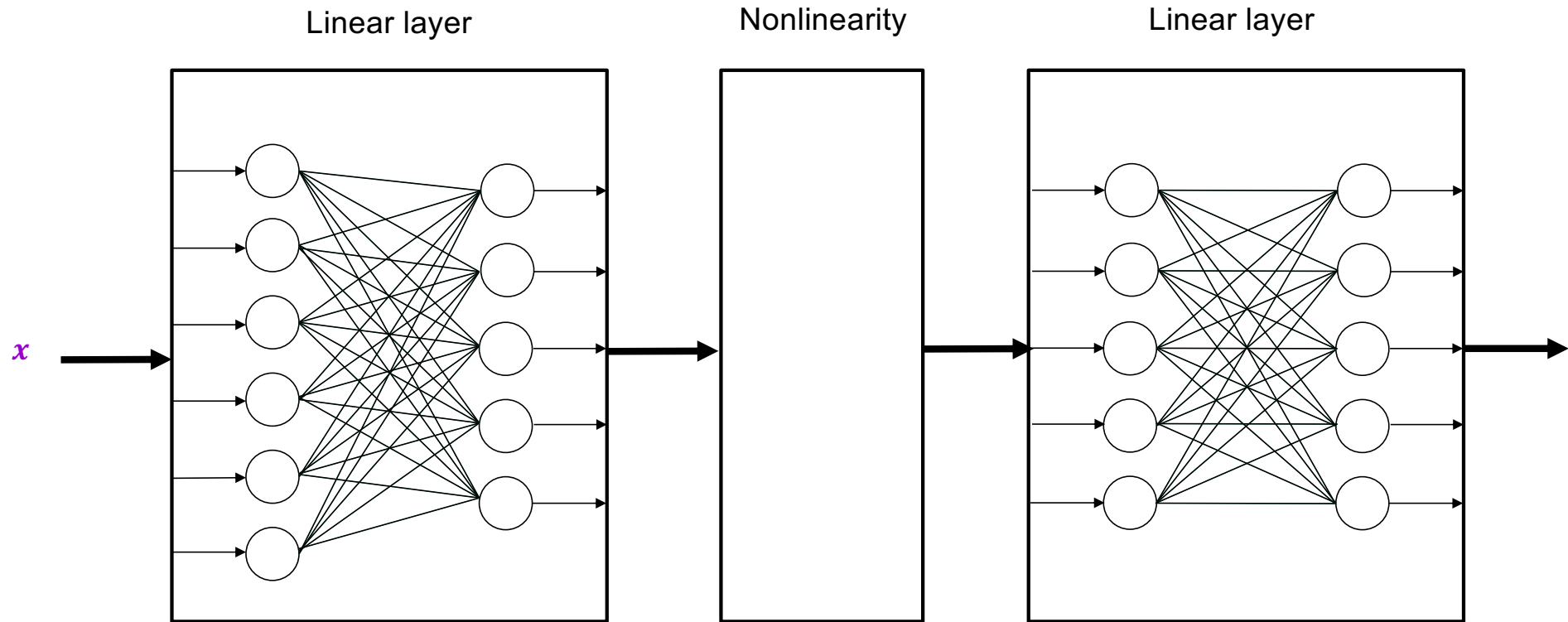


Let's add more layers!

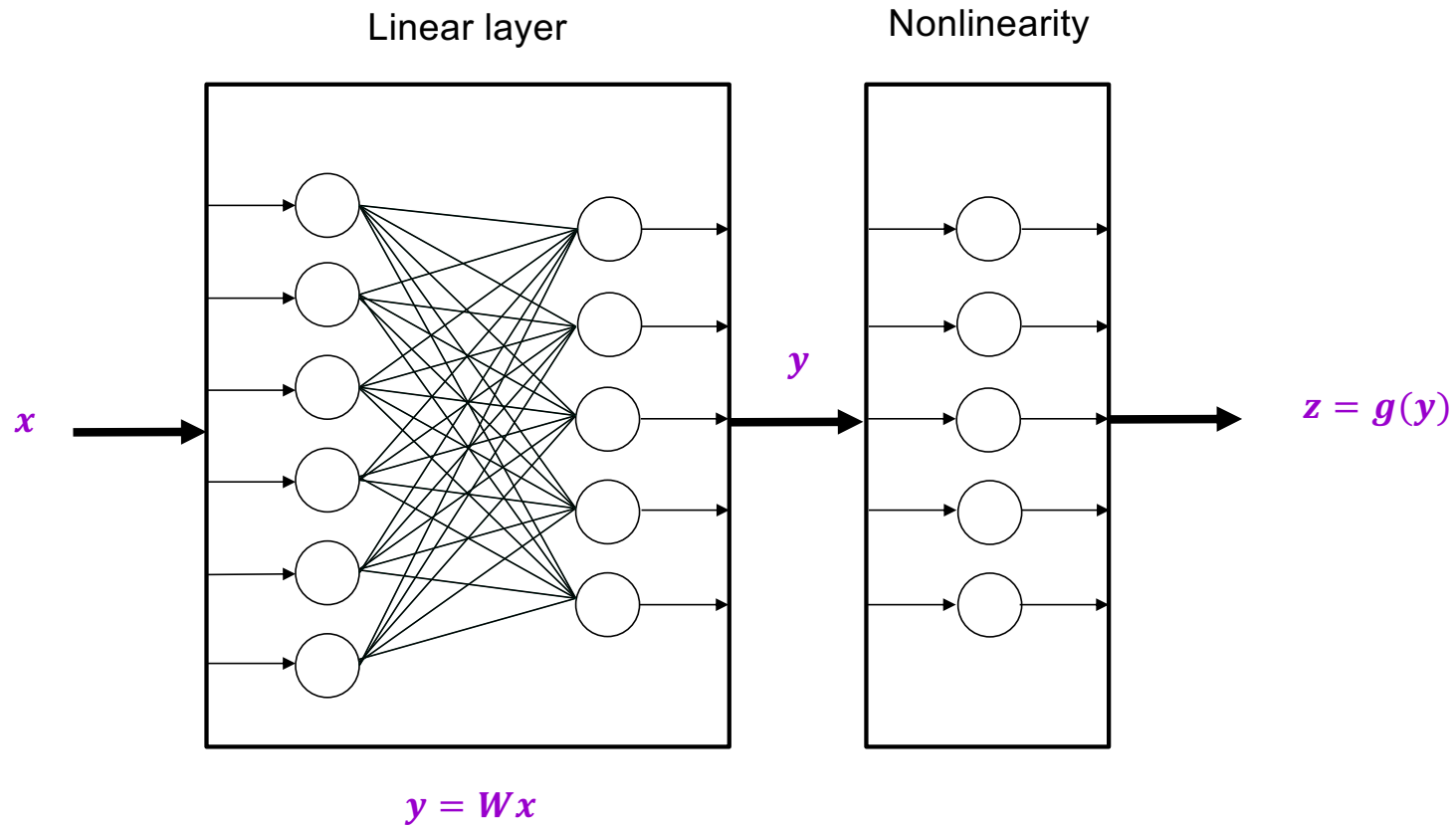
From linear units to multi-layer networks



From linear units to multi-layer networks



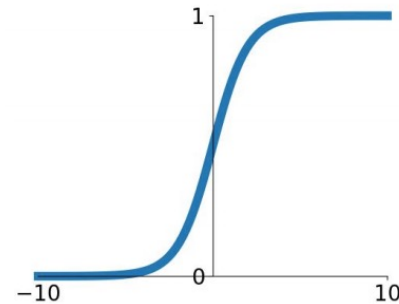
Element-wise nonlinearities (activation functions)



Element-wise nonlinearities (activation functions)

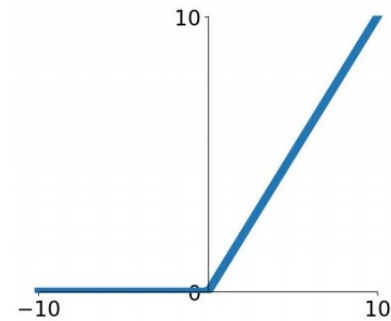
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



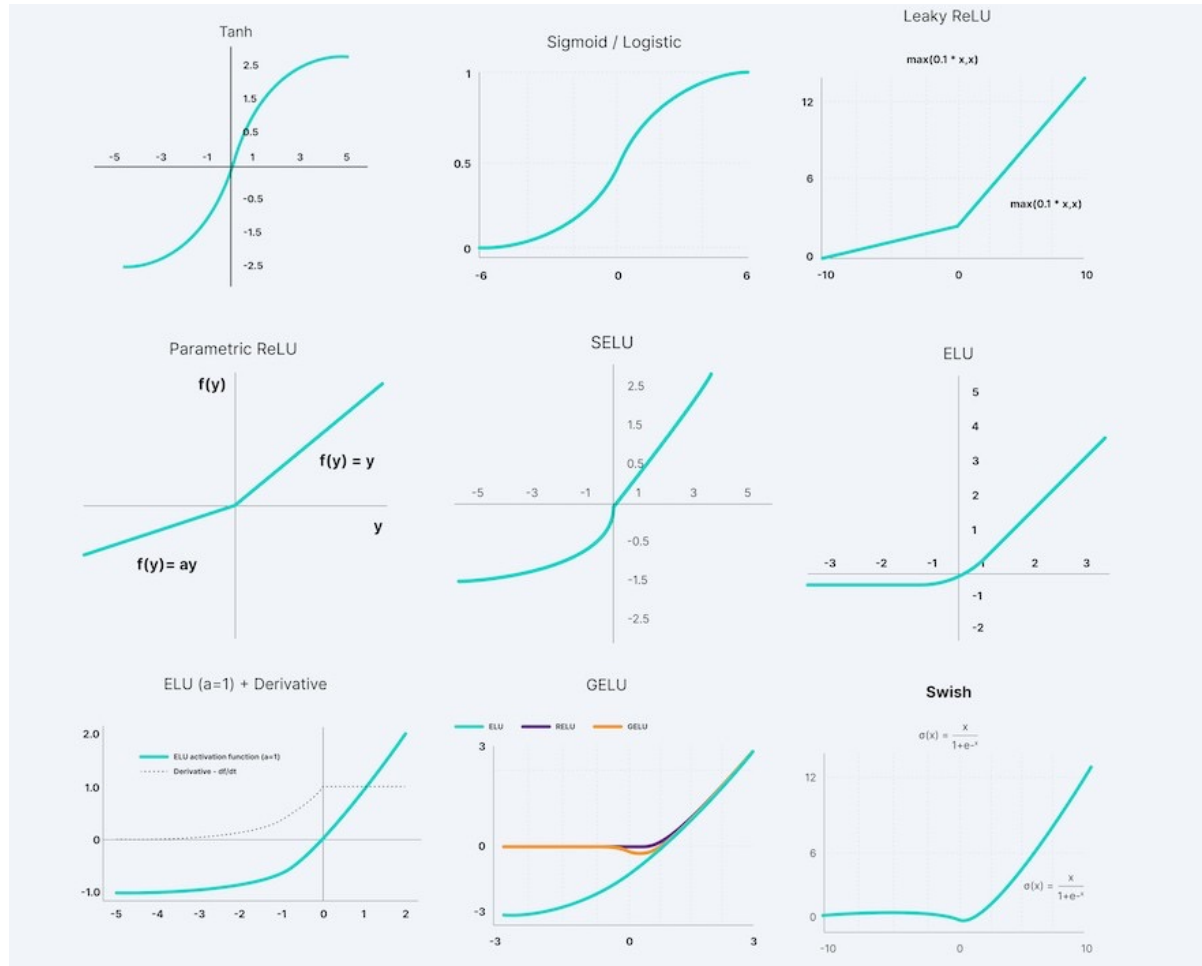
ReLU

$$\max(0, x)$$



Source: [Stanford 231n](#)

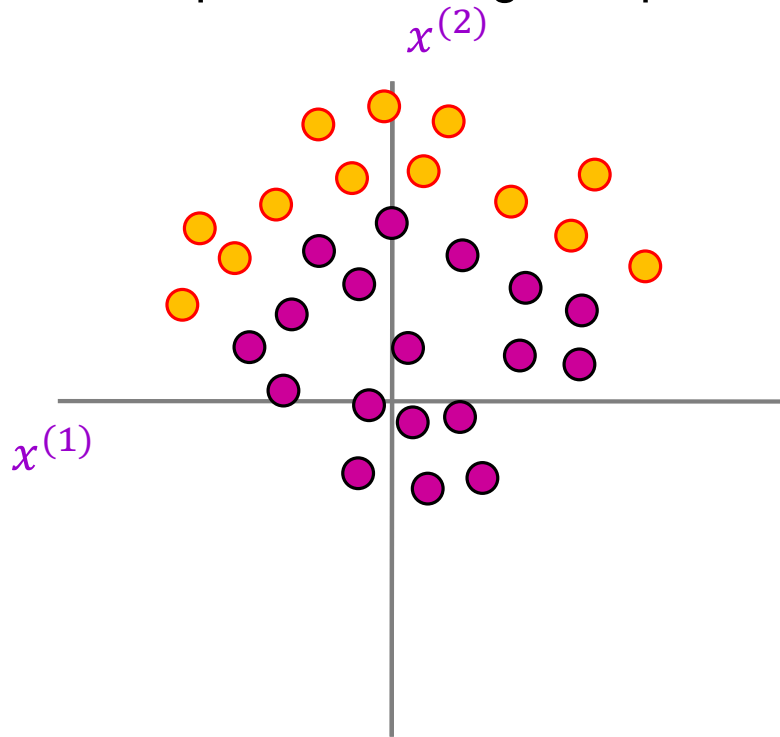
Element-wise nonlinearities (activation functions)



Source: [V7labs](https://v7labs.com/)

The power of ReLU

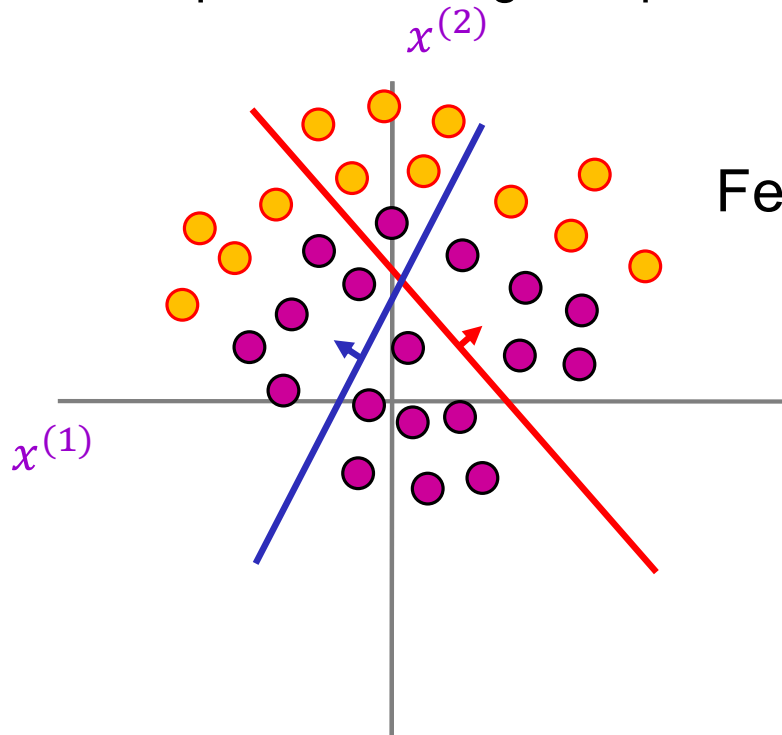
Points not linearly separable in original space



Source: [J. Johnson](#)

The power of ReLU

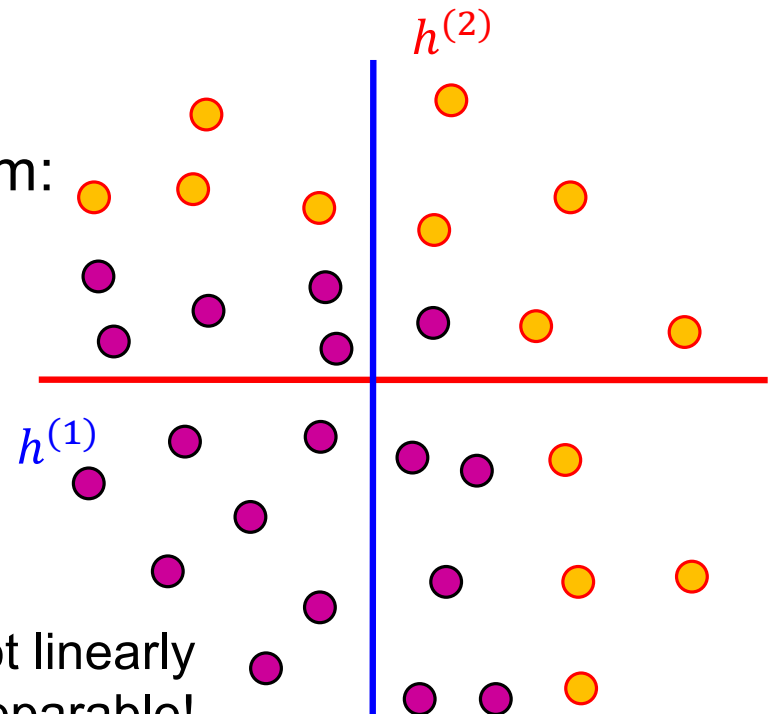
Points not linearly separable in original space



Consider a linear transform: $h = Wx + b$
Where x , h , b are 2-dimensional

Feature transform:

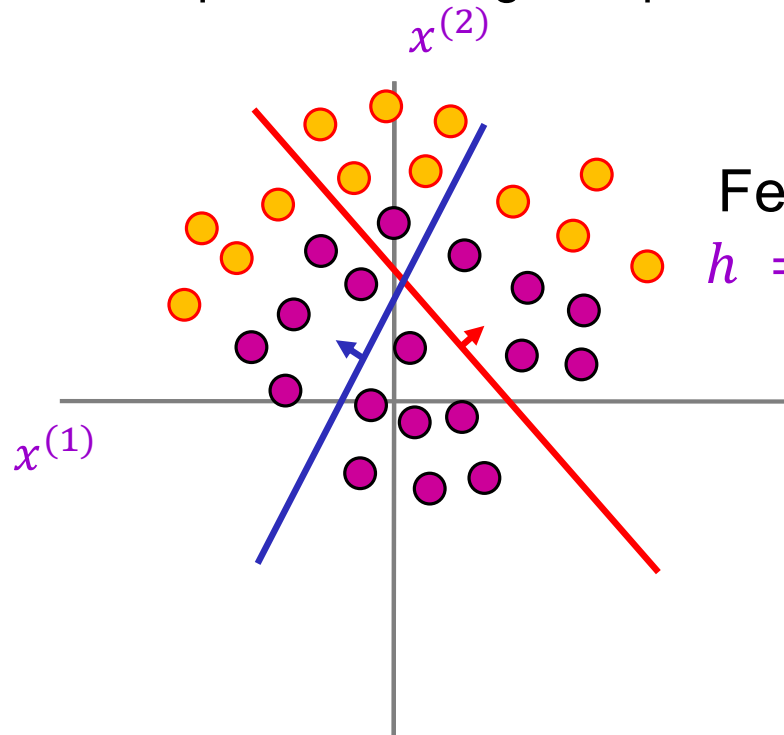
$$h = Wx + b$$



Still not linearly separable!

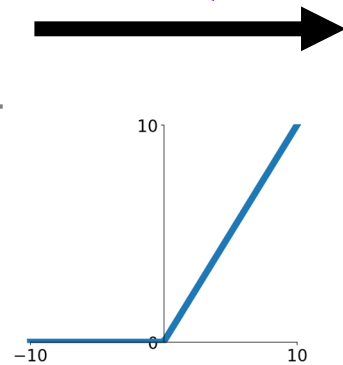
The power of ReLU

Points not linearly separable in original space



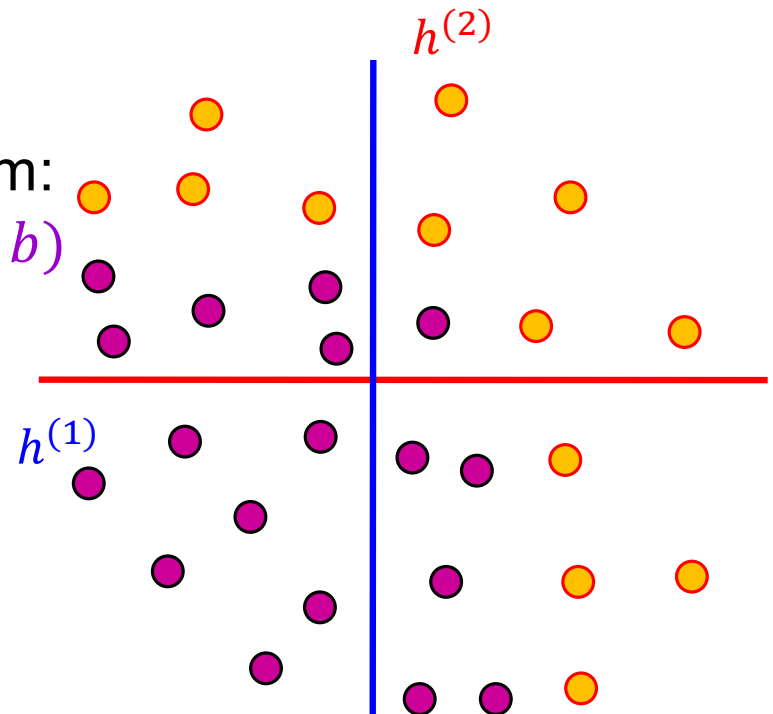
Feature transform:

$$h = \text{ReLU}(Wx + b)$$



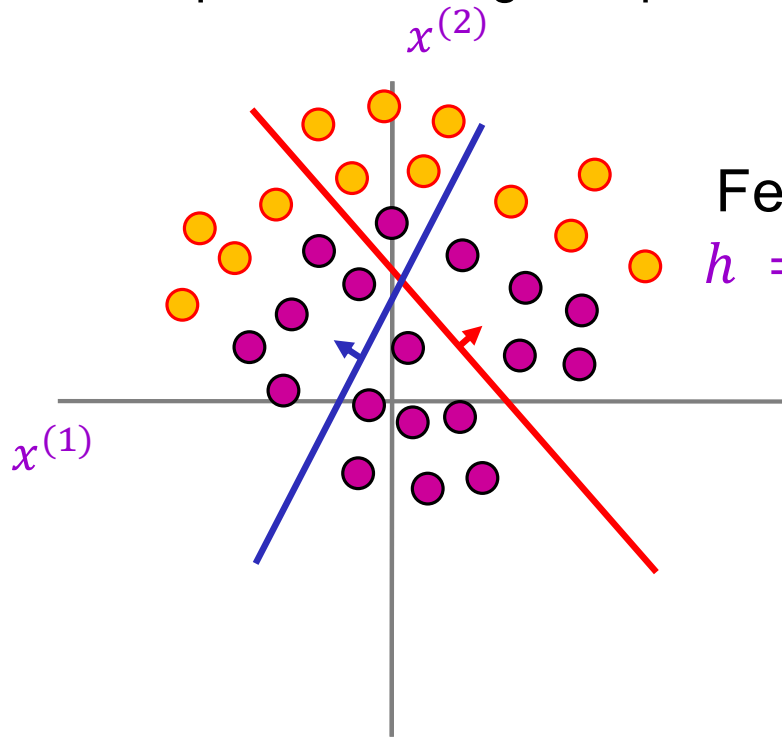
Let's add a nonlinearity:

$$h = \text{ReLU}(Wx + b) = \max(0, Wx + b)$$



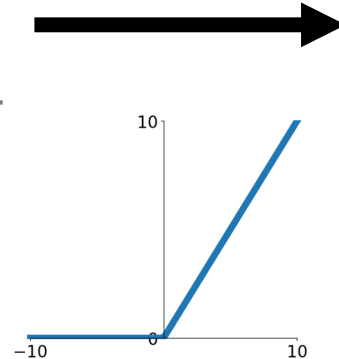
The power of ReLU

Points not linearly separable in original space



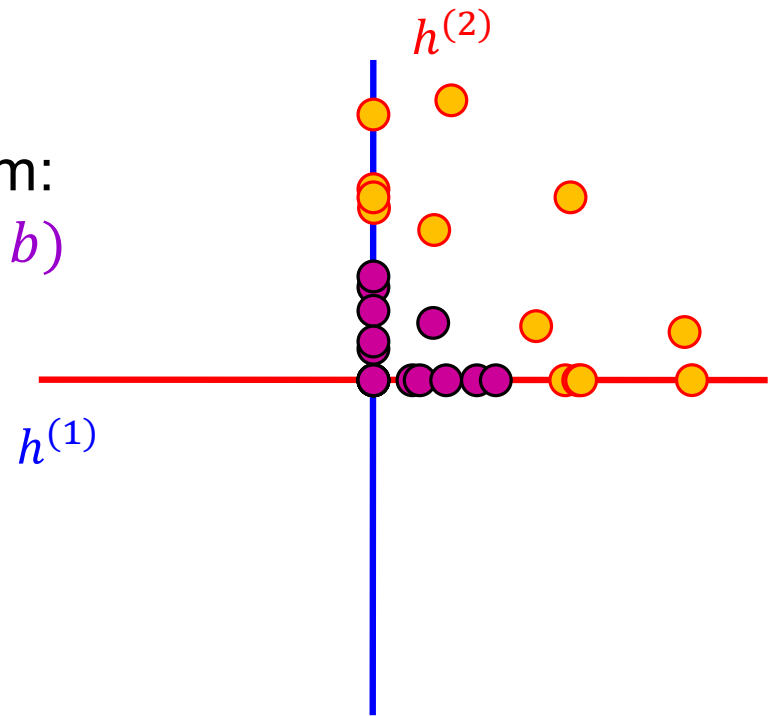
Feature transform:

$$h = \text{ReLU}(Wx + b)$$



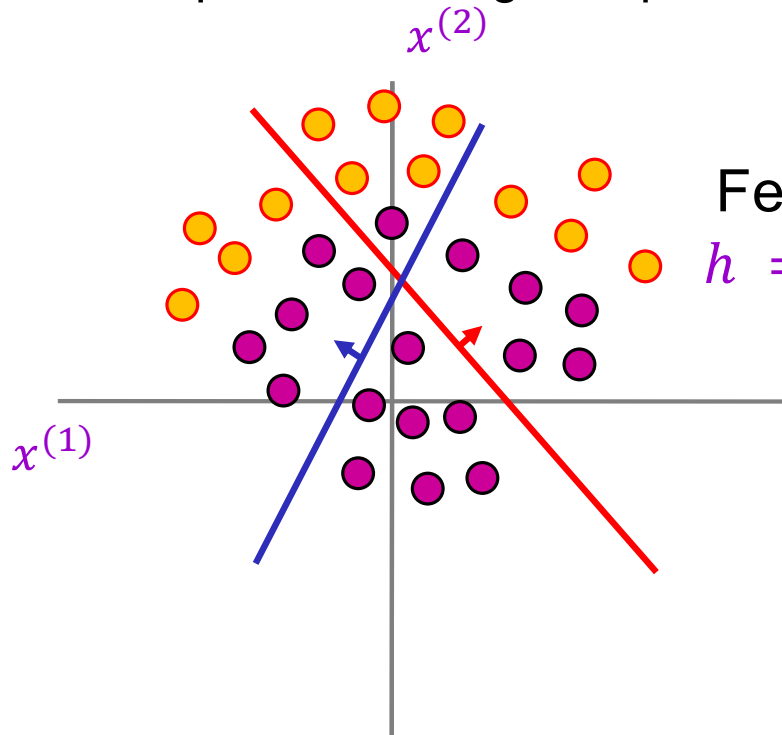
Let's add a nonlinearity:

$$h = \text{ReLU}(Wx + b) = \max(0, Wx + b)$$



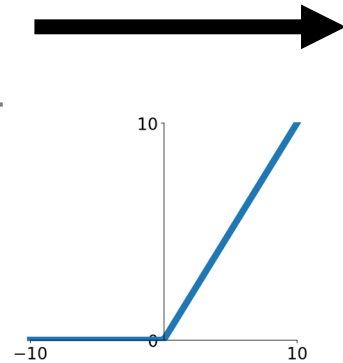
The power of ReLU

Points not linearly separable in original space



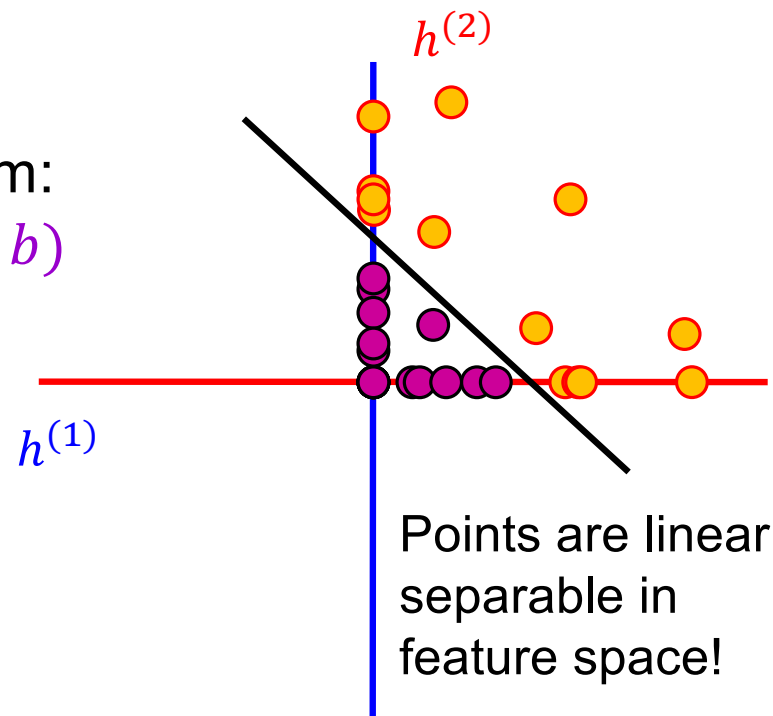
Feature transform:

$$h = \text{ReLU}(Wx + b)$$



Let's add a nonlinearity:

$$h = \text{ReLU}(Wx + b) = \max(0, Wx + b)$$

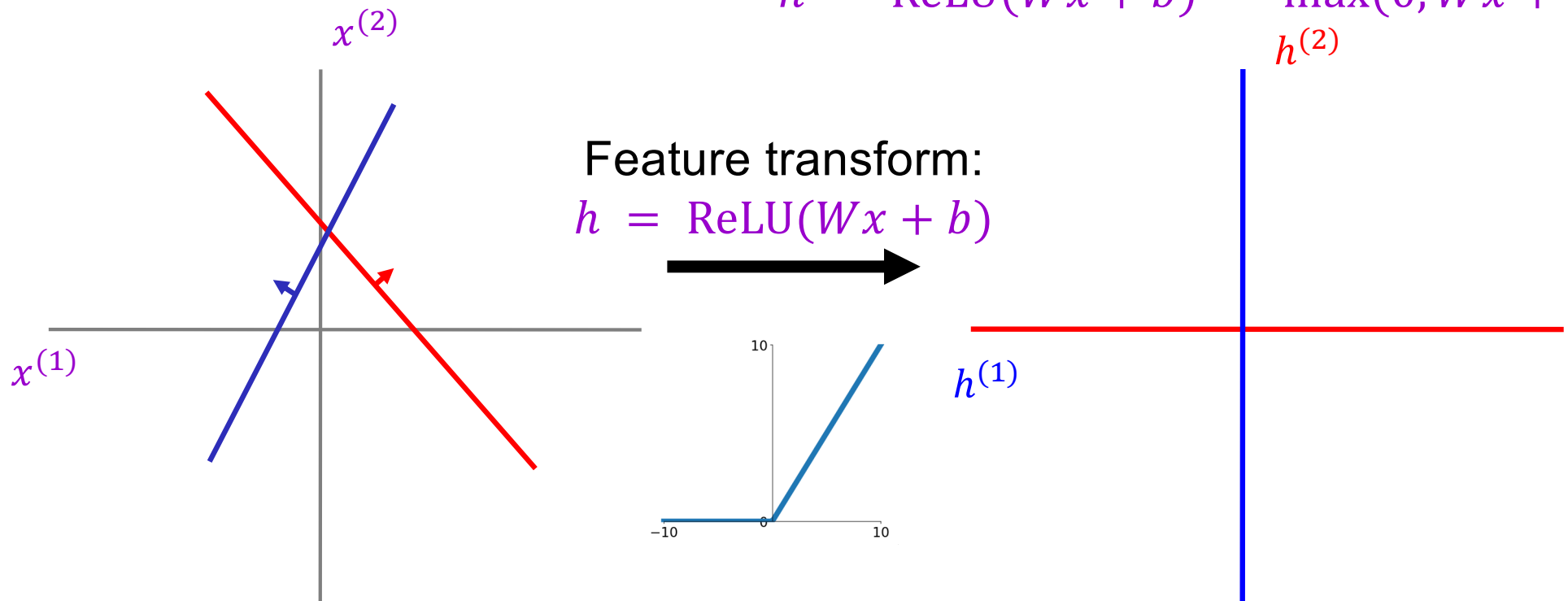


Points are linearly separable in feature space!

The power of ReLU

Let's add a nonlinearity:

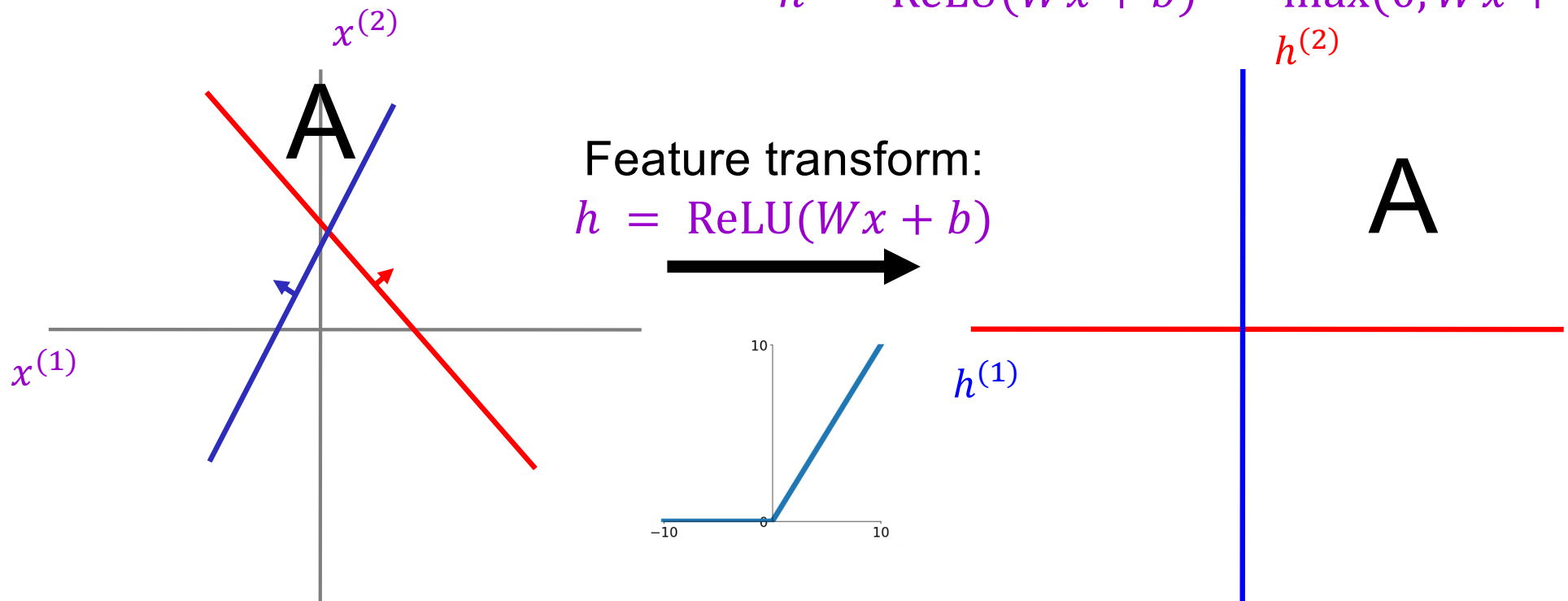
$$h = \text{ReLU}(Wx + b) = \max(0, Wx + b)$$



The power of ReLU

Let's add a nonlinearity:

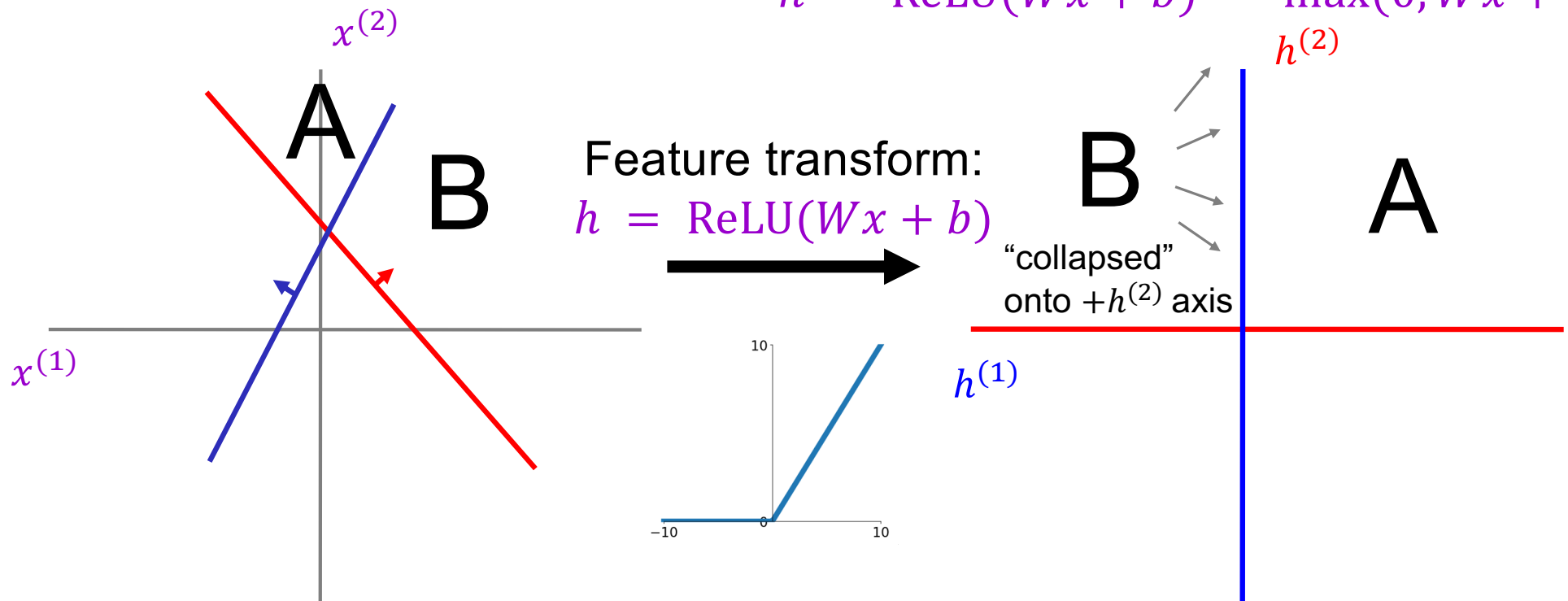
$$h = \text{ReLU}(Wx + b) = \max(0, Wx + b)$$



The power of ReLU

Let's add a nonlinearity:

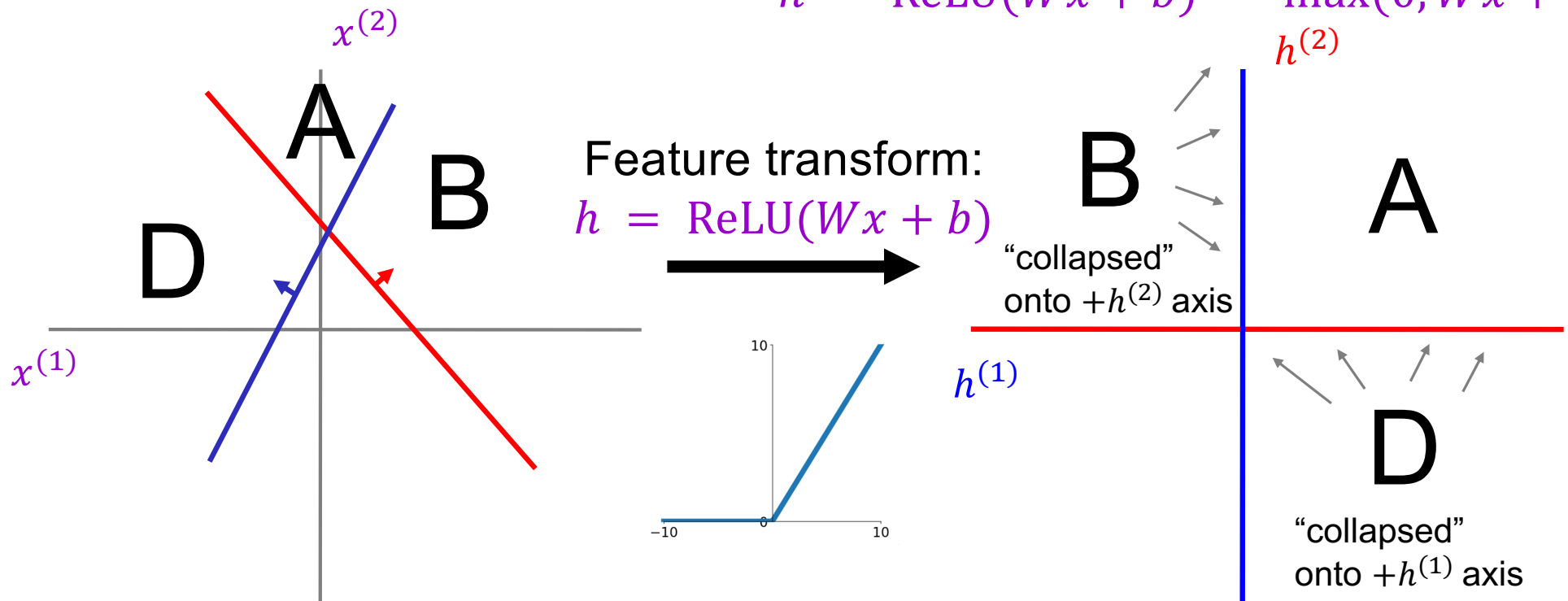
$$h = \text{ReLU}(Wx + b) = \max(0, Wx + b)$$



The power of ReLU

Let's add a nonlinearity:

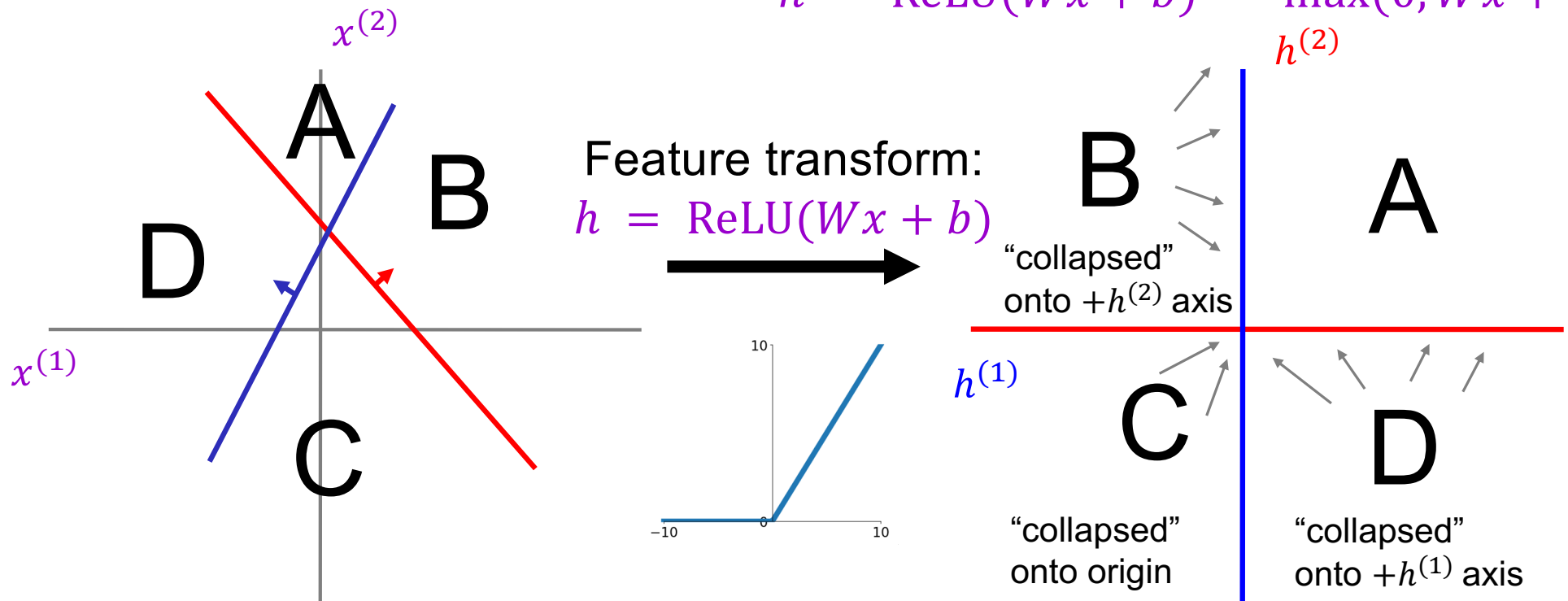
$$h = \text{ReLU}(Wx + b) = \max(0, Wx + b)$$



The power of ReLU

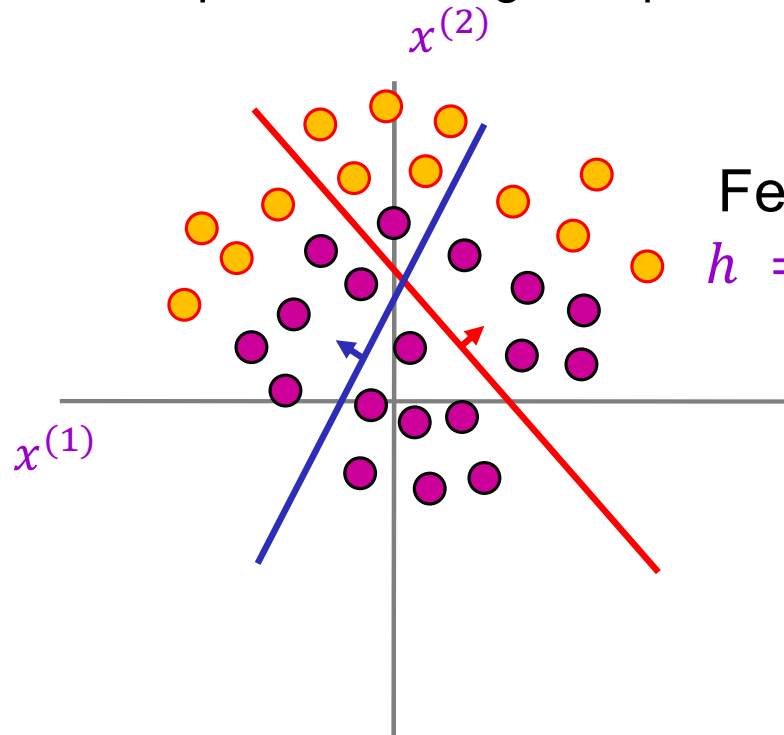
Let's add a nonlinearity:

$$h = \text{ReLU}(Wx + b) = \max(0, Wx + b)$$



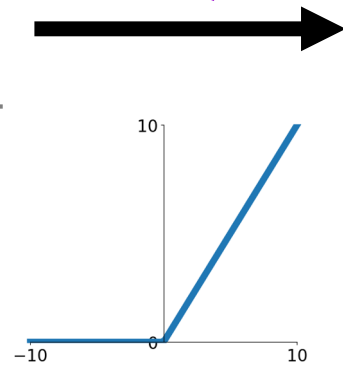
The power of ReLU

Points not linearly separable in original space



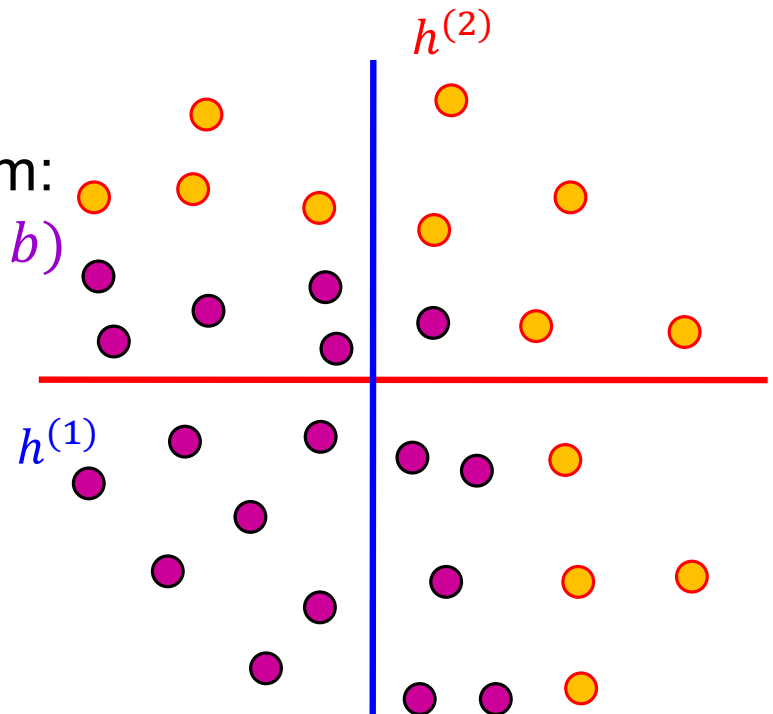
Feature transform:

$$h = \text{ReLU}(Wx + b)$$



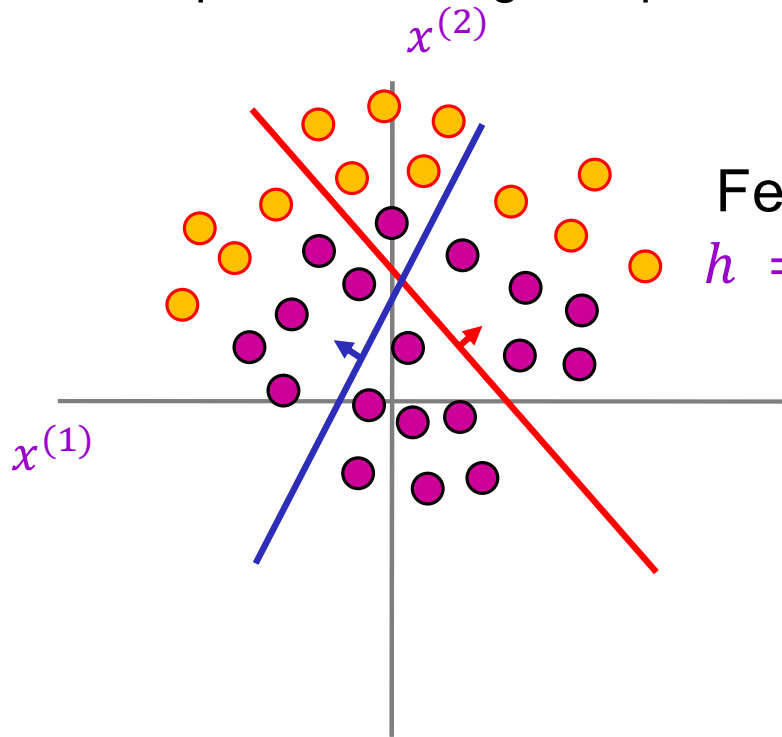
Let's add a nonlinearity:

$$h = \text{ReLU}(Wx + b) = \max(0, Wx + b)$$



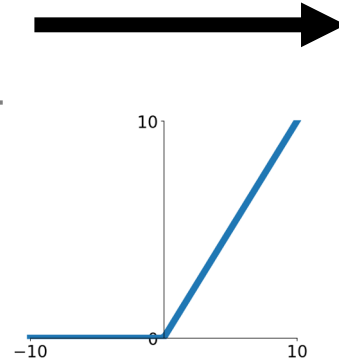
The power of ReLU

Points not linearly separable in original space



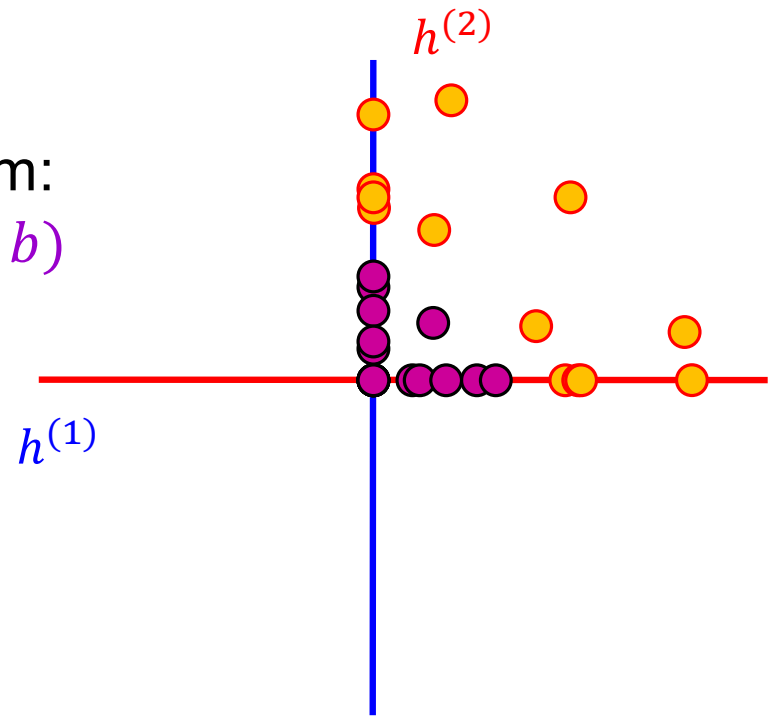
Feature transform:

$$h = \text{ReLU}(Wx + b)$$



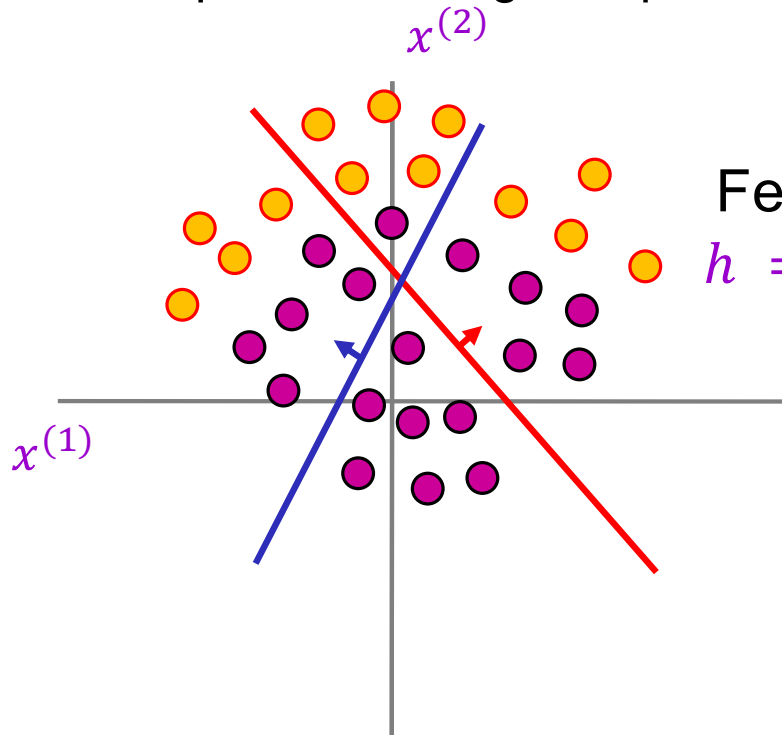
Let's add a nonlinearity:

$$h = \text{ReLU}(Wx + b) = \max(0, Wx + b)$$



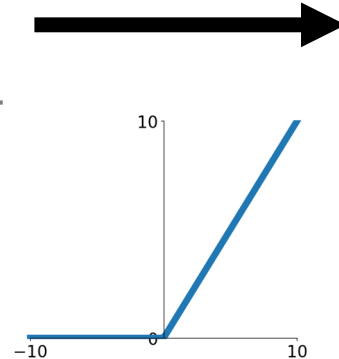
The power of ReLU

Points not linearly separable in original space



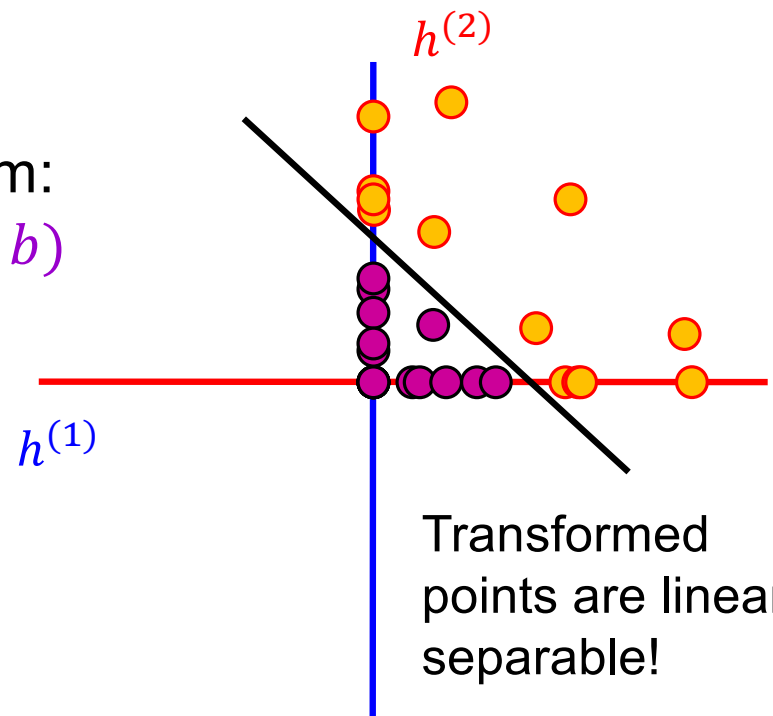
Feature transform:

$$h = \text{ReLU}(Wx + b)$$



Let's add a nonlinearity:

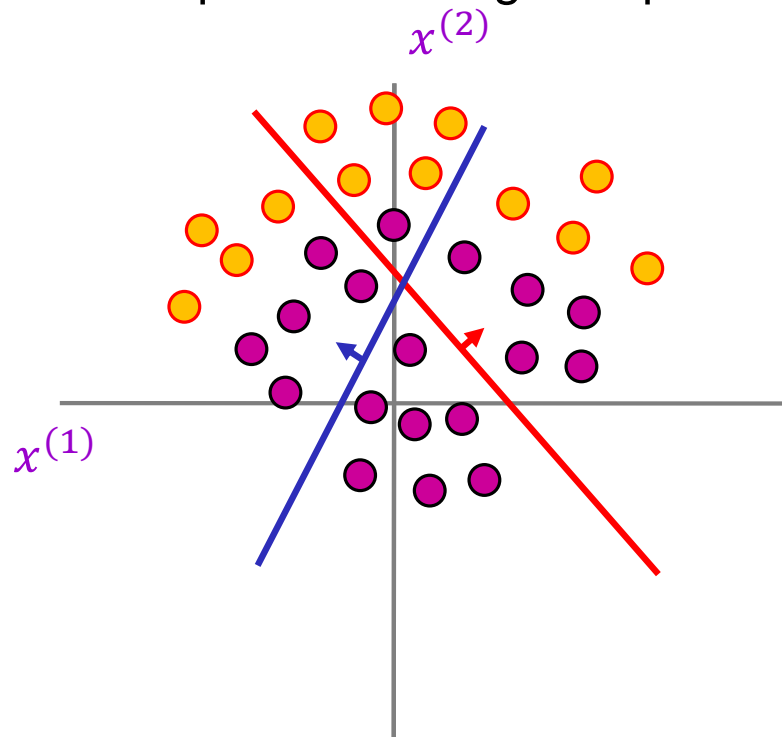
$$h = \text{ReLU}(Wx + b) = \max(0, Wx + b)$$



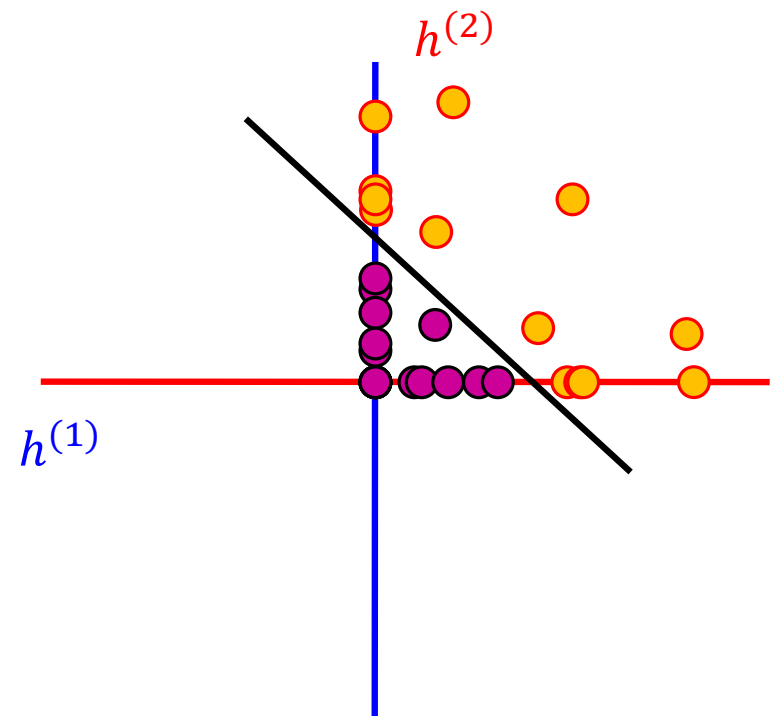
Transformed points are linearly separable!

The power of ReLU

Points not linearly separable in original space

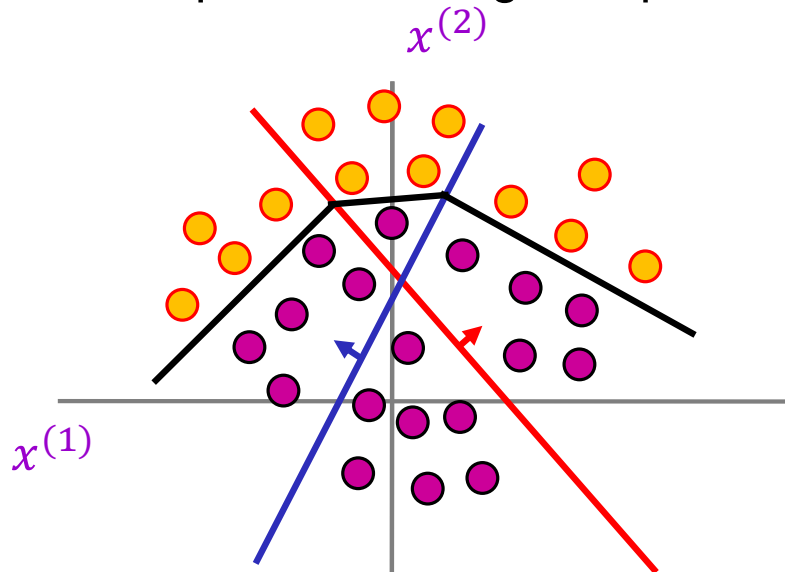


Let's trace the boundary back to the original space



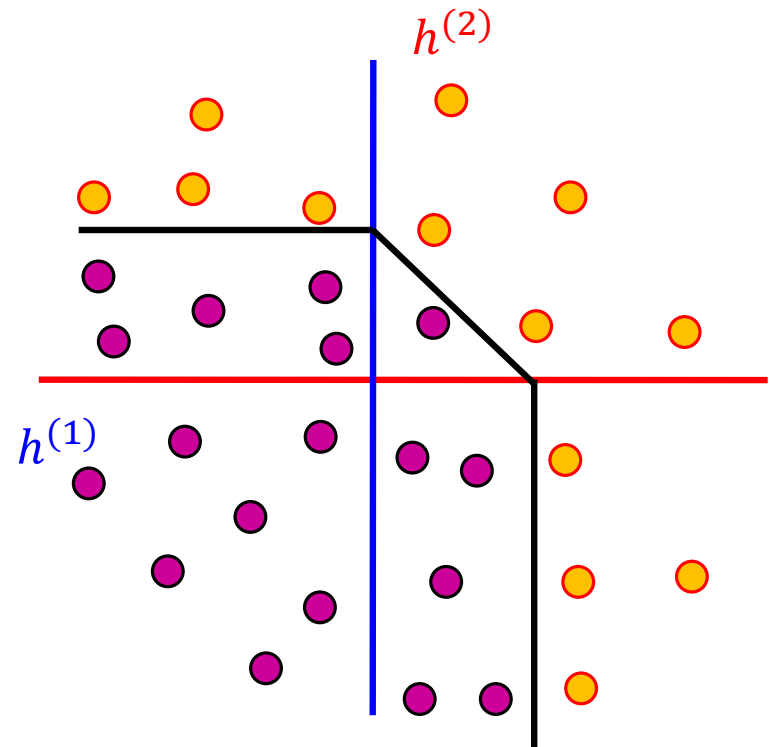
The power of ReLU

Points not linearly separable in original space

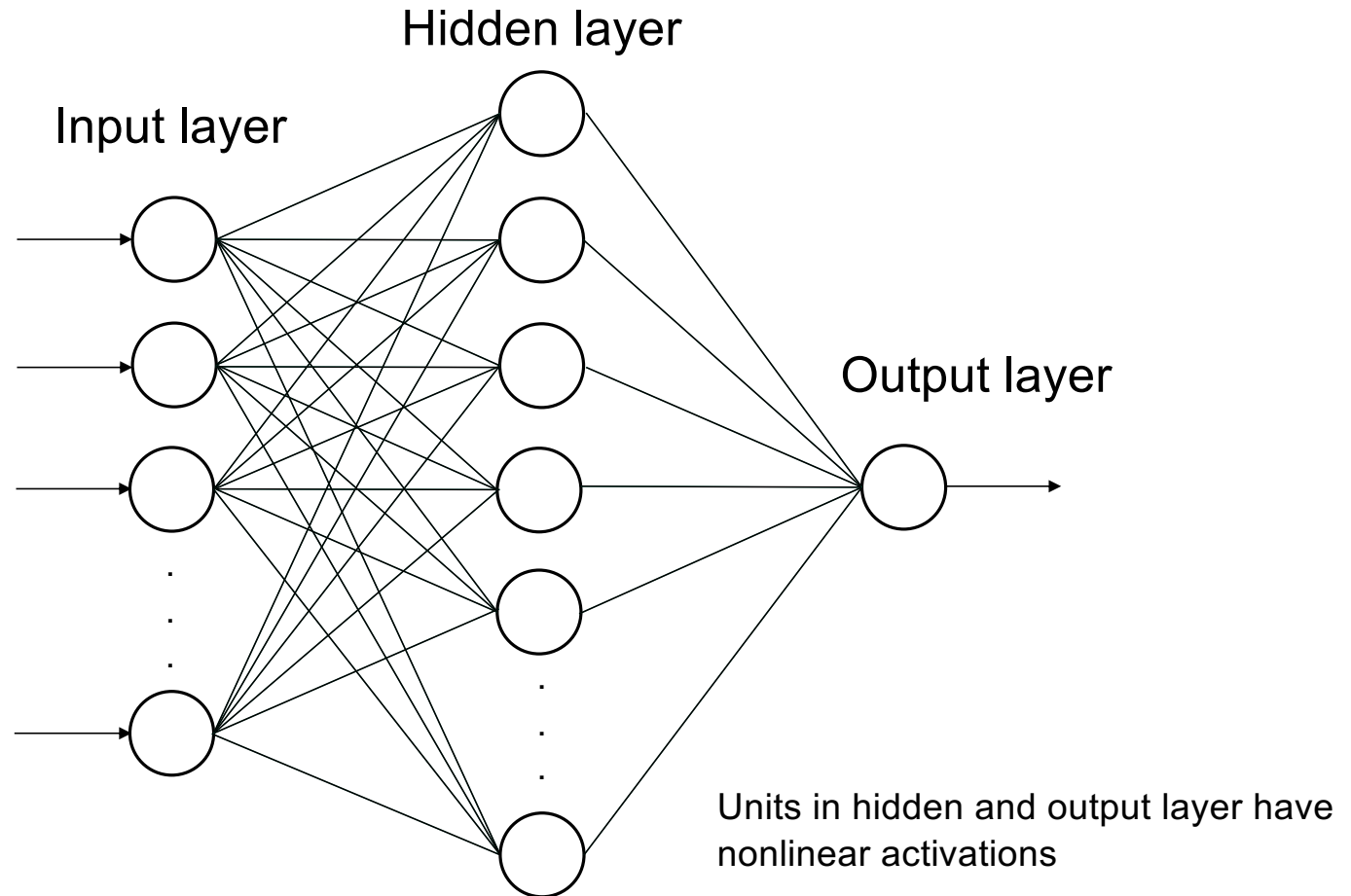


Linear boundary in transformed space gives nonlinear boundary in original space!

Let's trace the boundary back to the original space

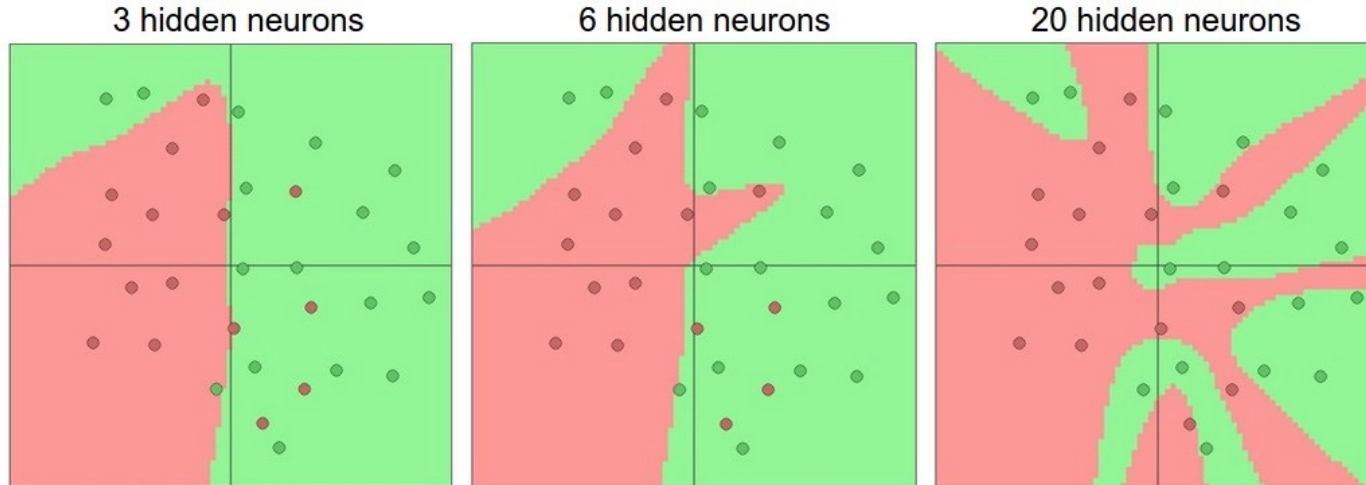


Two-layer neural network (for binary classification)



Expressiveness of two-layer networks

- How complex can we make the decision boundary in a two-layer network?
 - The bigger the hidden layer, the more complex the model
- A two-layer network is a [universal function approximator](#)
 - But the hidden layer may need to be huge



[Figure source](#)

Two-layer networks for images

- Recall: linear classifier weights are per-class templates



Two-layer networks for images

- What about a two-layer network?

First layer: bank of templates



Source: [J. Johnson](#)

Two-layer networks for images

- What about a two-layer network?

First layer: bank of templates



Templates can correspond to different *modes* of the data

Two-layer networks for images

- What about a two-layer network?

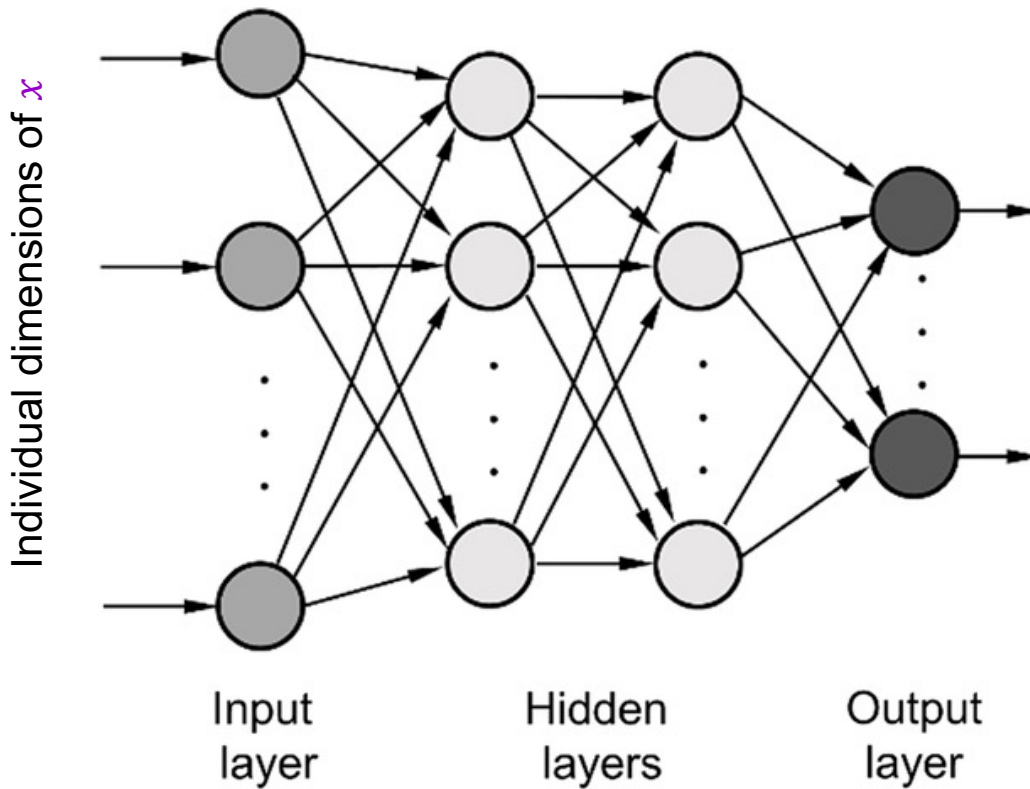
First layer: bank of templates

Second layer: weights for combining template activations



Not all templates are interpretable — it's a “distributed” representation

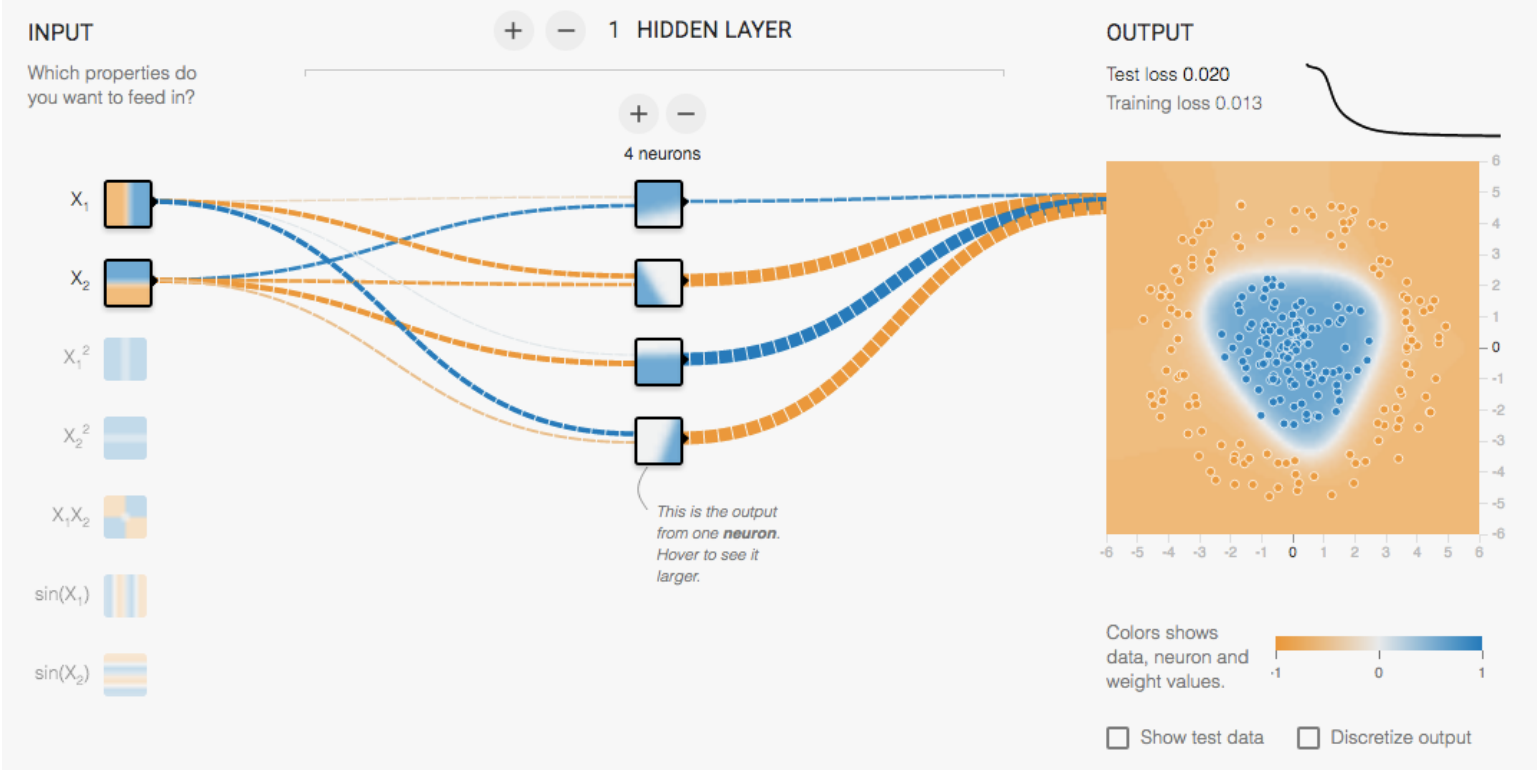
Neural networks beyond two layers



Output:

$$g_L(W_L \dots g_2(W_2 g_1(W_1 x)) \dots)$$

Multi-Layer network demo



<http://playground.tensorflow.org/>

Overview

- Multi-layer networks
 - Linear layers
 - Activation functions
- Hyperparameter search, generalization

Recall: Supervised learning outline

1. **Collect data and labels**
2. **Specify model:** select model class and loss function
3. **Train model:** find the parameters of the model that minimize the empirical loss on the training data

This involves
hyperparameters that
affect the generalization
ability of the trained model

Hyperparameters

- Regularization constant λ in SVM optimization

$$\min_w \frac{\lambda}{2} \|w\|^2 + \sum_{i=1}^n \max[0, 1 - y_i w^T x_i]$$

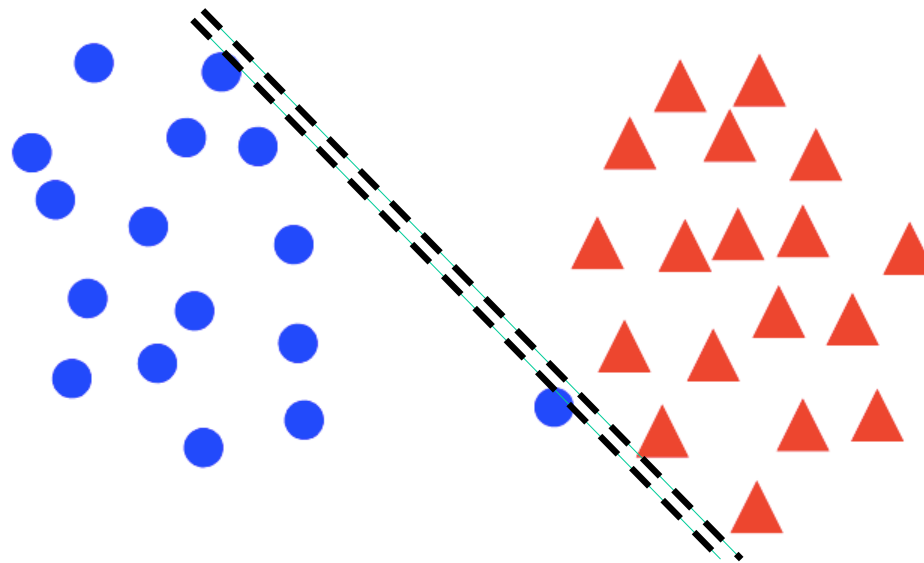
- Tradeoff between margin and classification errors

Hyperparameters

- Regularization constant λ in SVM optimization

$$\min_w \frac{\lambda}{2} \|w\|^2 + \sum_{i=1}^n \max[0, 1 - y_i w^T x_i]$$

- Tradeoff between margin and classification errors



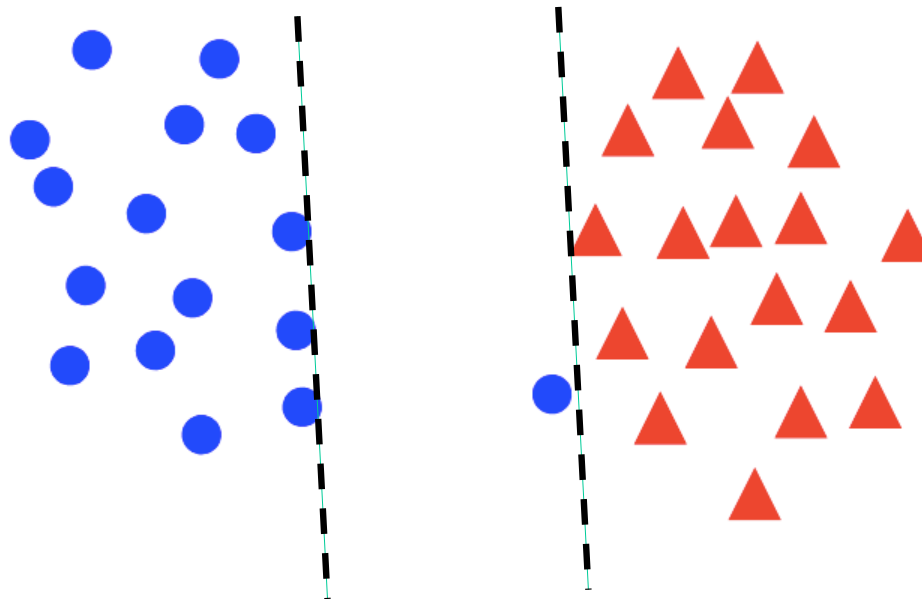
[Source](#)

Hyperparameters

- Regularization constant λ in SVM optimization

$$\min_w \frac{\lambda}{2} \|w\|^2 + \sum_{i=1}^n \max[0, 1 - y_i w^T x_i]$$

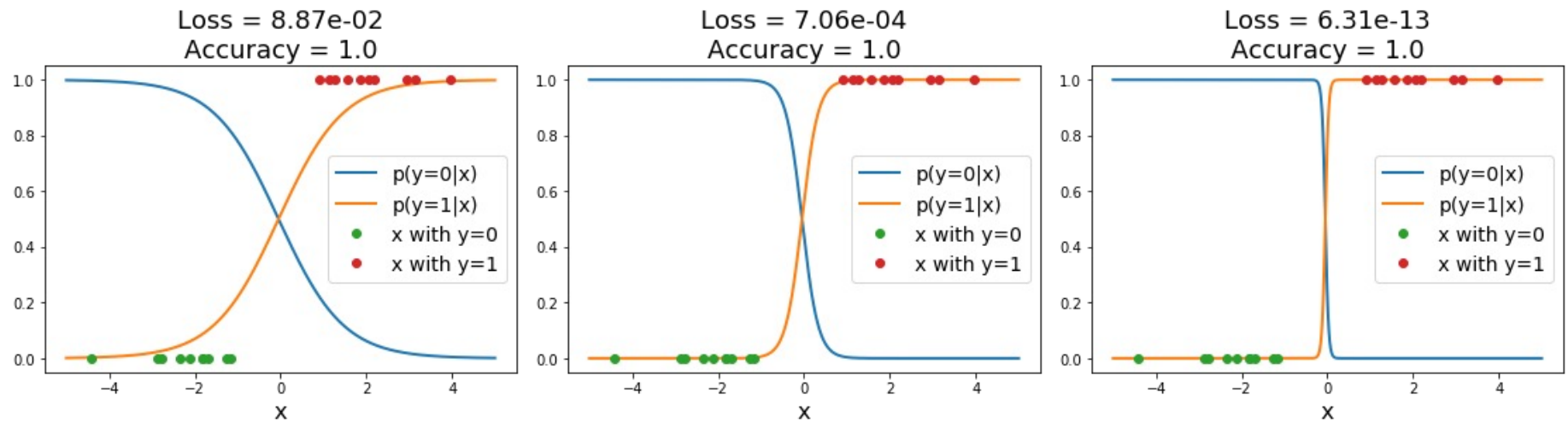
- Tradeoff between margin and classification errors



[Source](#)

Hyperparameters

- Regularization constant λ in logistic regression
 - Preventing the classifier from getting over-confident

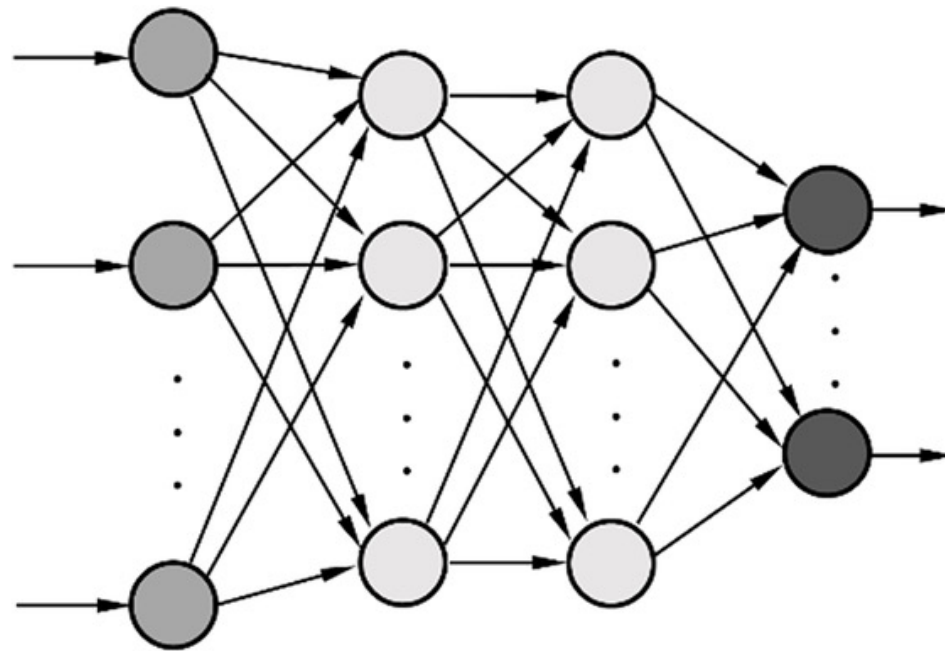


Sigmoid classifier, logistic loss

Source: [J. Johnson](#)

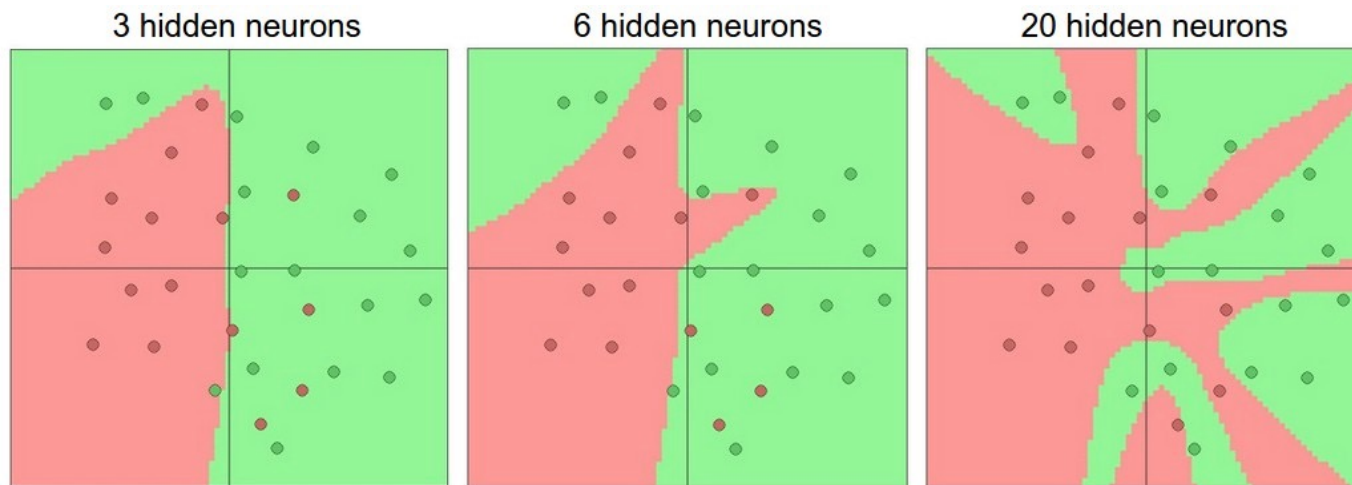
Hyperparameters in multi-layer networks

- Number of layers, number of units per layer



Hyperparameters in multi-layer networks

- Number of layers, number of units per layer

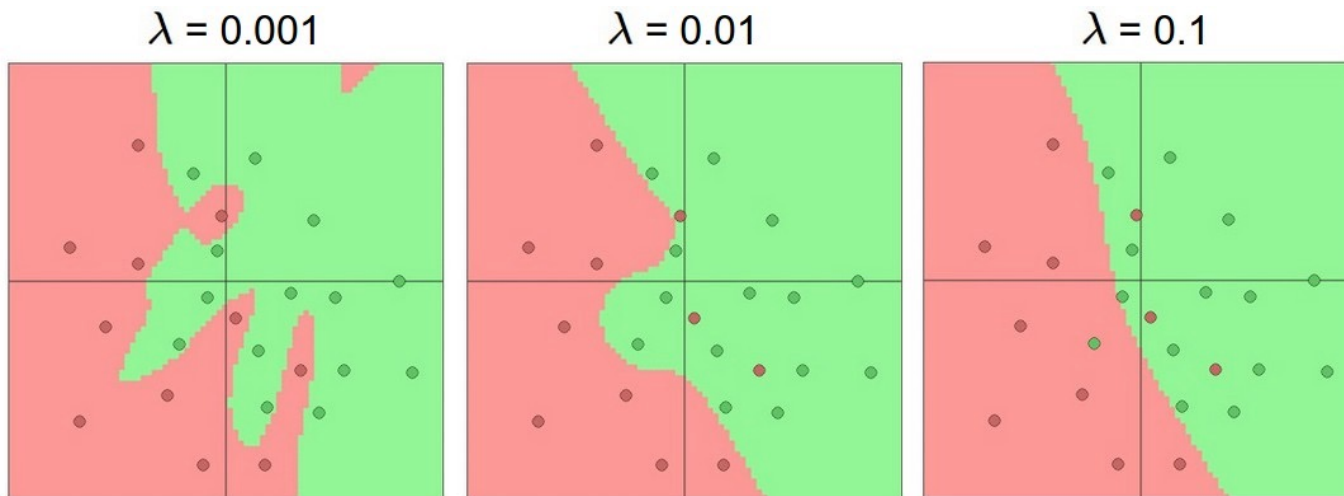


Number of hidden units in a two-layer network

Source: [Stanford 231n](#)

Hyperparameters in multi-layer networks

- Number of layers, number of units per layer
- Type of activation functions
- Type of loss function
- Regularization constant



Source: [Stanford 231n](#)

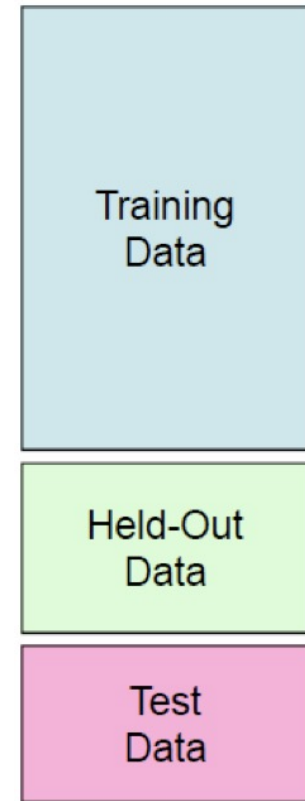
Hyperparameters in multi-layer networks

- Number of layers, number of units per layer
- Type of activation functions
- Type of loss function
- Regularization constant
- SGD settings: learning rate schedule, number of epochs, minibatch size, etc.

- Our hyperparameter choices affect the expressiveness of the model and its ability to generalize to new data
 - To measure generalization ability, we need to track both *training* and *test (or held-out) error* of the model

Hyperparameter search: The short story

- For each combination of hyperparameter choices, iterate:
 - Instantiate model
 - Learn parameters on the **training data**
 - Measure accuracy on the **held-out** or **validation data**
- Finally, measure accuracy on the **test data**
- You should avoid peeking at the test set during hyperparameter search since it is supposed to represent *never before seen data*



The mysteries of generalization

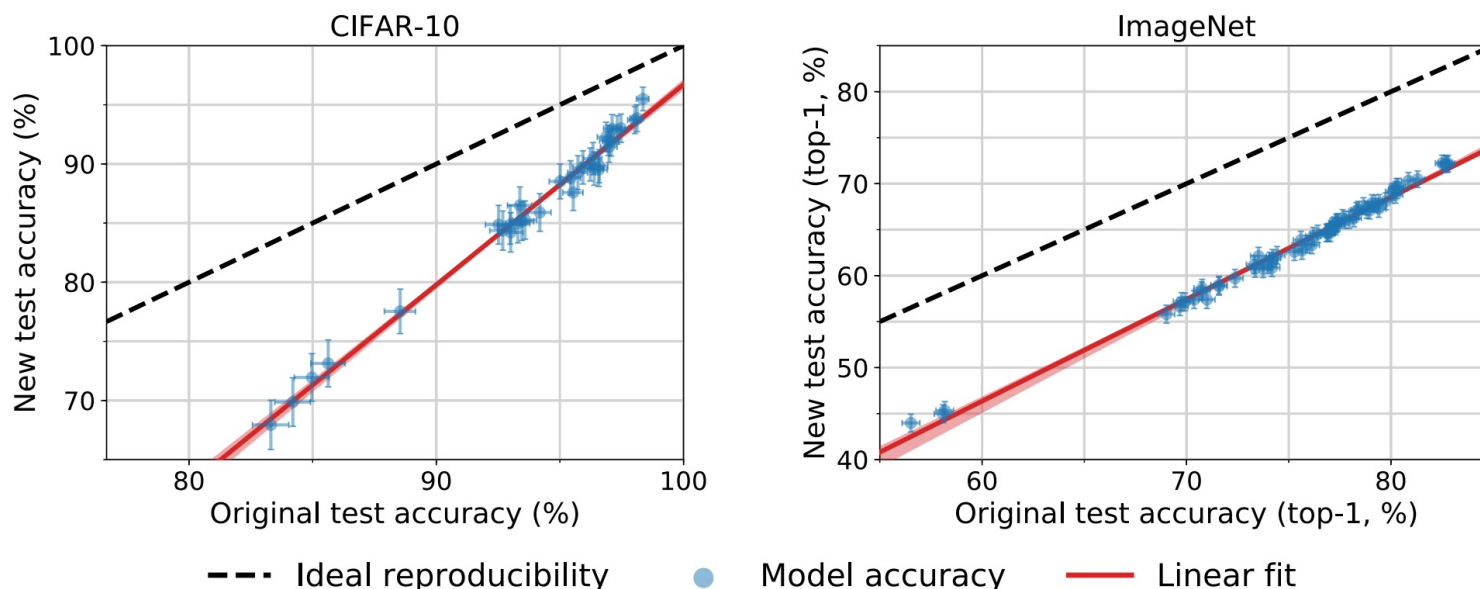


Figure 1. Model accuracy on the original test sets vs. our new test sets. Each data point corresponds to one model in our testbed (shown with 95% Clopper-Pearson confidence intervals). The plots reveal two main phenomena: (i) There is a significant drop in accuracy from the original to the new test sets. (ii) The model accuracies closely follow a linear function with slope *greater* than 1 (1.7 for CIFAR-10 and 1.1 for ImageNet). This means that every percentage point of progress on the original test set translates into more than one percentage point on the new test set. The two plots are drawn so that their aspect ratio is the same, i.e., the slopes of the lines are visually comparable. The red shaded region is a 95% confidence region for the linear fit from 100,000 bootstrap samples.

B. Recht et al. [Do ImageNet classifiers generalize to ImageNet?](#) ICML 2019

See also: C. Zhang et al. [Understanding deep learning \(still\) requires rethinking generalization.](#) Comm. ACM, 2021

Ben Recht [post 1](#), [post 2](#), [post 3](#)