Agenda

Linear Algebra

Calculus

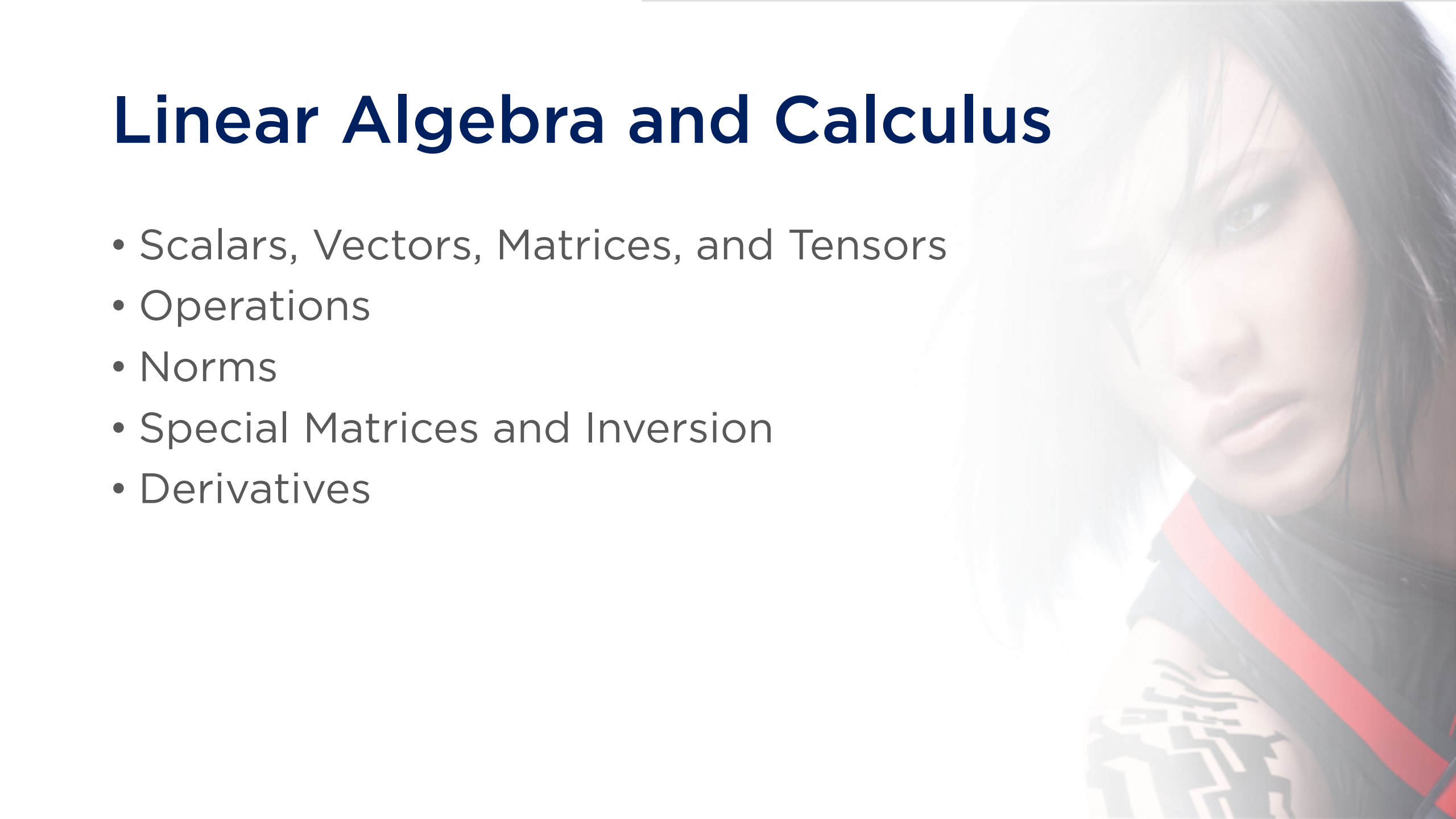
A Brief History of Neural Networks

Neural Networks in Spark

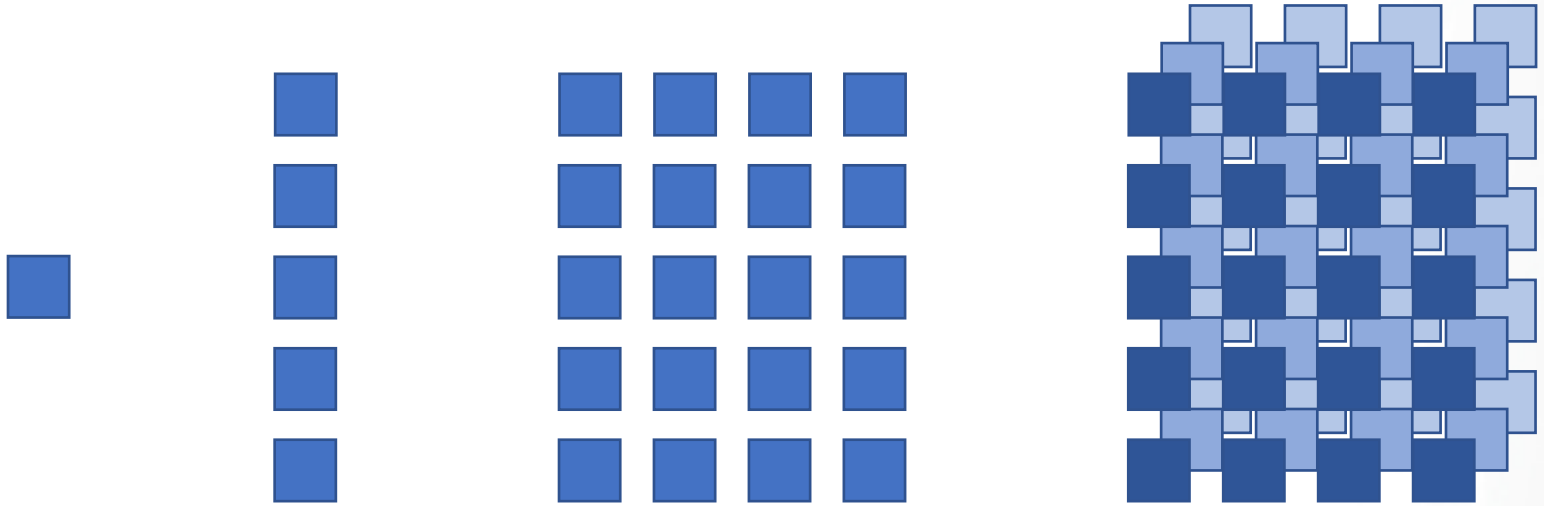
Math Is Fun

Linear Algebra and Calculus

- Scalars, Vectors, Matrices, and Tensors
- Operations
- Norms
- Special Matrices and Inversion
- Derivatives



Scalars, Vectors, Matrices, and Tensors



Scalar Vector Matrix Tensor

Operations

- Addition
- Transpose
- Multiplication

Addition

Scalar: 2 + 3 = 5

Vector: $\begin{bmatrix}1\\2\end{bmatrix} + \begin{bmatrix}3\\4\end{bmatrix} = \begin{bmatrix}4\\6\end{bmatrix}$

Matrix: $\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} + \begin{bmatrix}5 & 6\\7 & 8\end{bmatrix} = \begin{bmatrix}6 & 8\\10 & 12\end{bmatrix}$

Transpose

Scalar: $2^T = 2$

Vector: $\begin{bmatrix}1\\2\end{bmatrix}^T = \begin{bmatrix}1 & 2\end{bmatrix}$

Matrix: $\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}^T = \begin{bmatrix}1 & 3\\2 & 4\end{bmatrix}$

Multiplication

Two types of multiplication

- Inner Product
- Outer Product

Inner Product

- Dimension reducing

Scalar: (2)(3) = 6

Vector: $\begin{bmatrix}1 & 2\end{bmatrix}\begin{bmatrix}3\\4\end{bmatrix} = [(1)(3)+(2)(4)] = [11]$

Matrix/Vector: $\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}\begin{bmatrix}5\\6\end{bmatrix} = \begin{bmatrix}(1)(5)+(2)(6)\\(3)(5)+(4)(6)\end{bmatrix}$

Matrix: $\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}\begin{bmatrix}5 & 6\\7 & 8\end{bmatrix} = \begin{bmatrix}(1)(5)+(2)(7) & (1)(6)+(2)(8)\\(3)(5)+(4)(7) & (3)(6)+(4)(8)\end{bmatrix}$

Outer Product

- Dimension increasing

Vector/Vector: $\begin{bmatrix}1 & 2\end{bmatrix} \otimes \begin{bmatrix}3\\4\end{bmatrix} = \begin{bmatrix}3 & 6\\4 & 8\end{bmatrix}$

Vector⊗Matrix→Tensor(3d)

Norms

Length of a vector

$$\|X\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} = \sqrt{X^T X}$$

$$\|X\|^2 = X^T X$$

Distance

$$\|X - Y\|$$

Identity and Inverses

Scalar: $(1)(5) = 5 \quad ab = 1 \rightarrow a = 1/b$

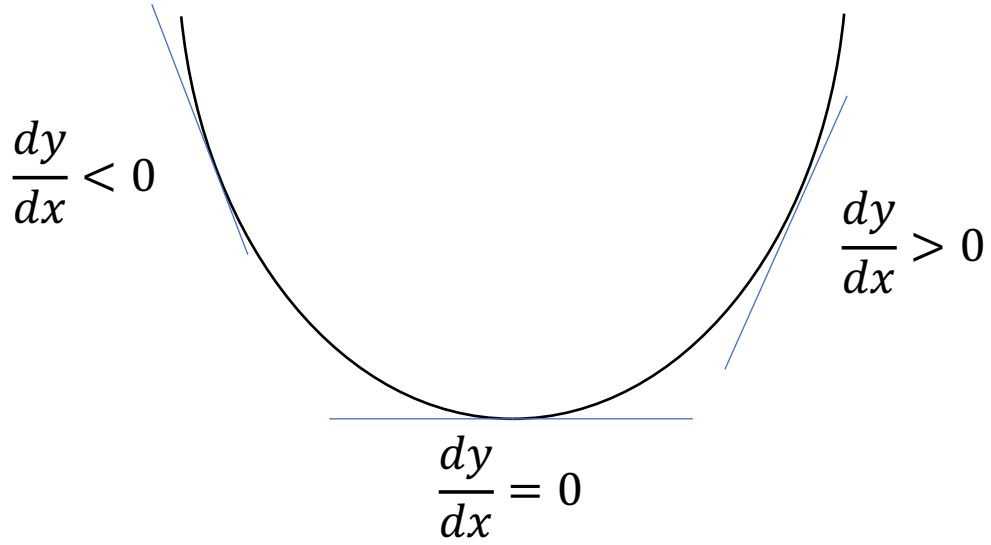
Matrices

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix}$$

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad IA = AI = A$$

$$AA^{-1} = I$$

Calculus



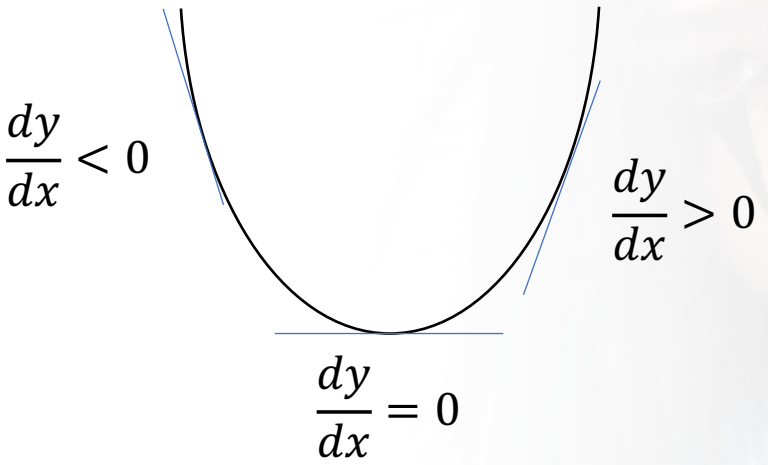
Slope is zero at minimum

Calculus

Gradient Decent

Find minimum of $f(x)$

```
pick x
while 0 < f(x)
    let Δx = −df(x)/dx
    update x = x + Δx
```



Calculus

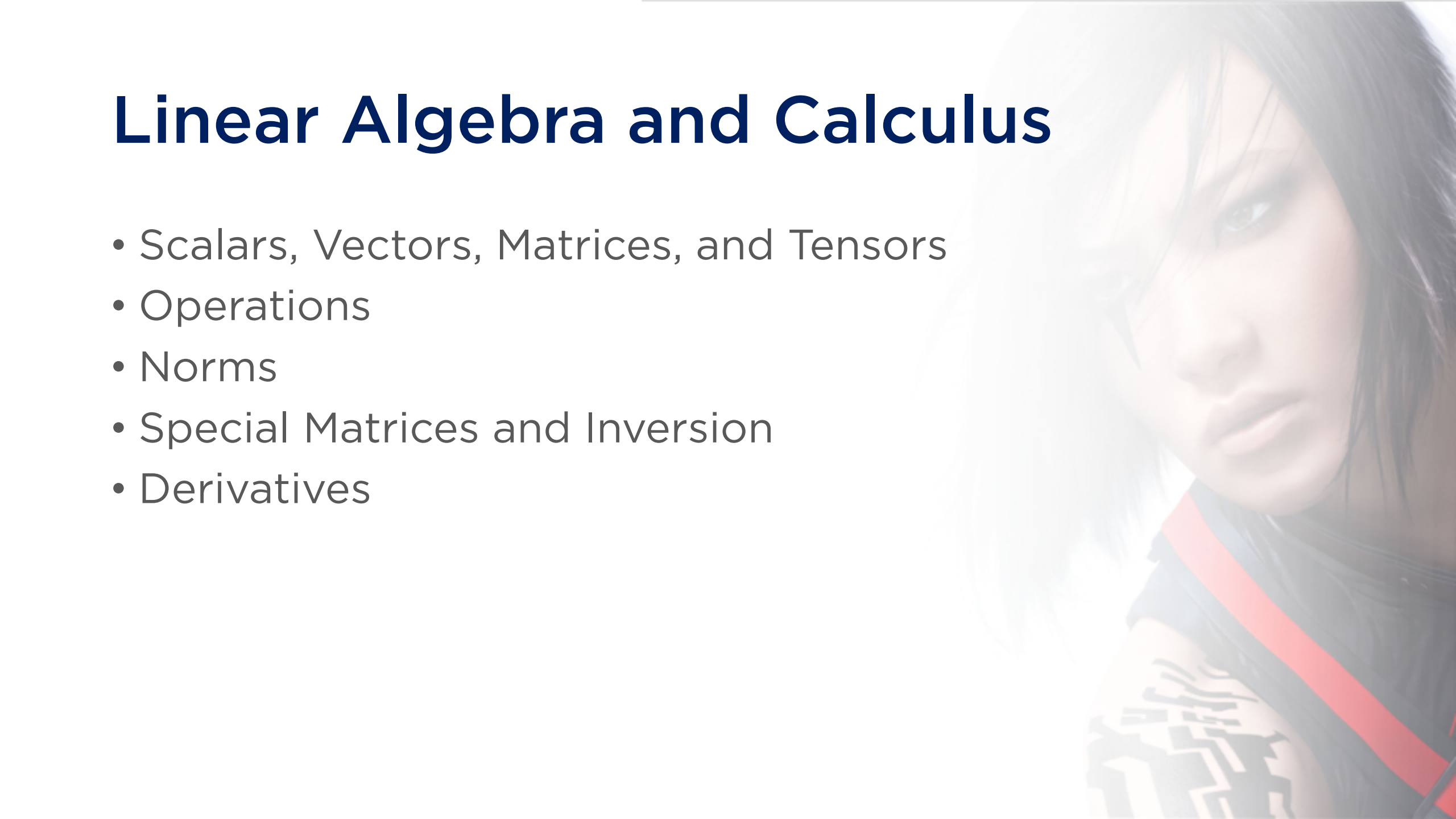
Chain Rule

$z = f(y) \qquad y = g(x)$

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$

Linear Algebra and Calculus

- Scalars, Vectors, Matrices, and Tensors
- Operations
- Norms
- Special Matrices and Inversion
- Derivatives

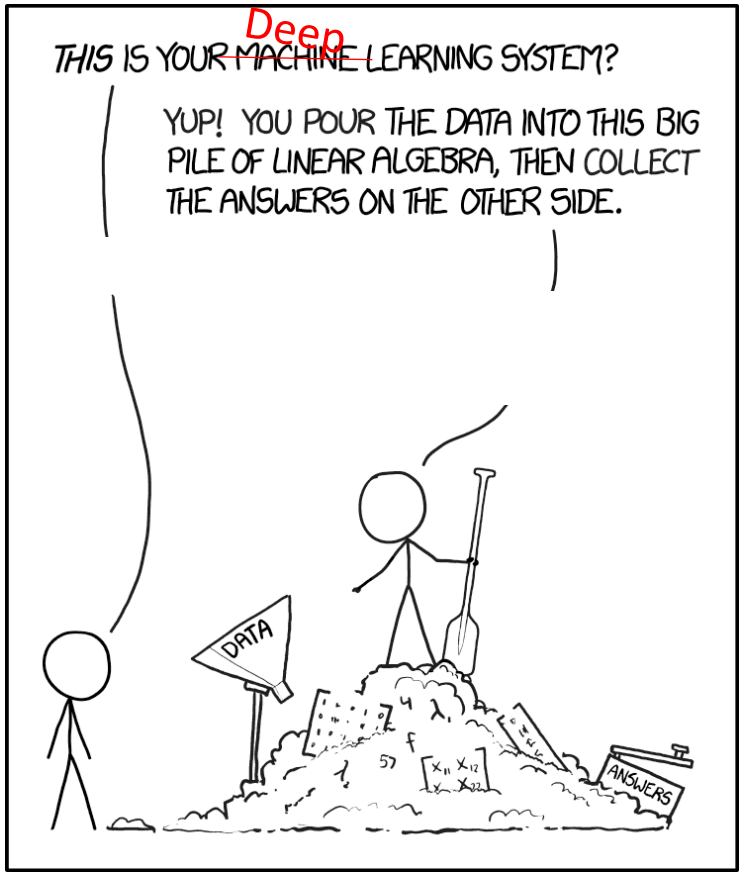


A Brief History of Neural Networks

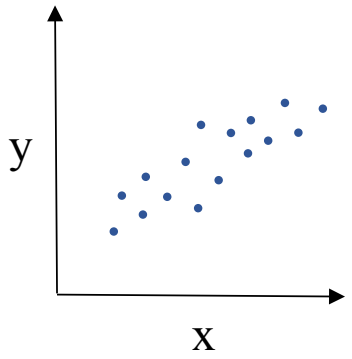
Neural Networks

- Linear Regression
- Logistic Regression
- Perceptrons
- Multilayered Perceptrons
- Support Vector Machines
- Deep Learning

After 120 Years of Research

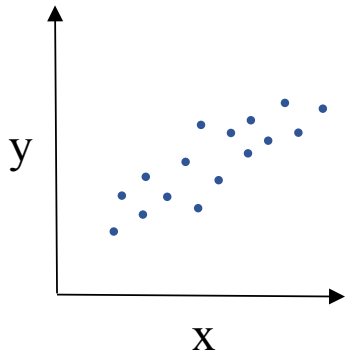


In The Beginning There Was Data



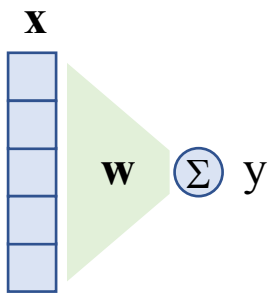
Data

Linear Regression (1894)



Data

$$y = \sum_{i=1}^{n} w_i x_i$$



$$y = \mathbf{w}^T\mathbf{x}$$

Mechanism

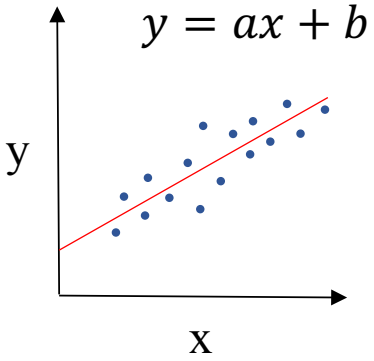
Minimize

$$\|\mathbf{y} - \mathrm{X}\mathbf{w}\|$$

Closed form solution for $\mathbf{w}$

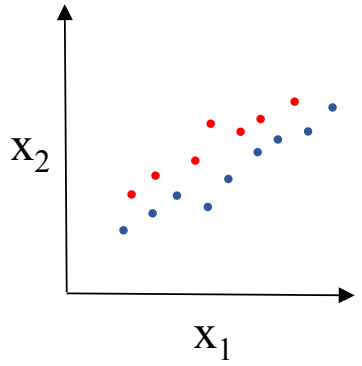
$$\mathbf{w} = (X^TX)^{-1}X\mathbf{y}$$

Learning



Model

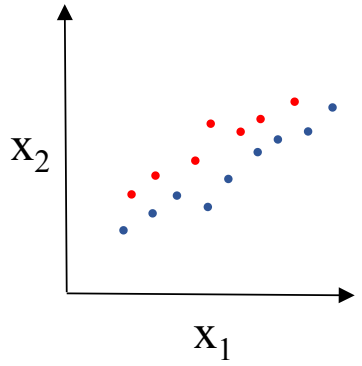
Logistic Regression (1958)



$y \in \{0,1\}$

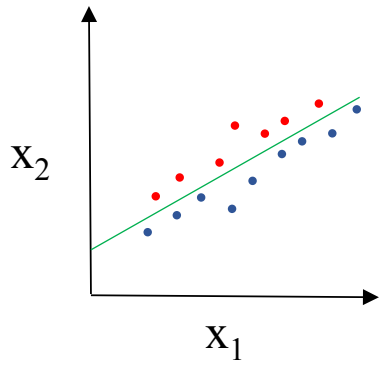
Data

Logistic Regression (1958)



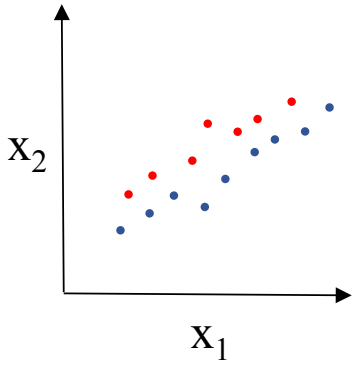
$y \in \{0,1\}$

Data



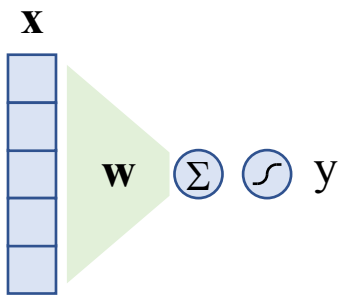
Model

Logistic Regression (1958)



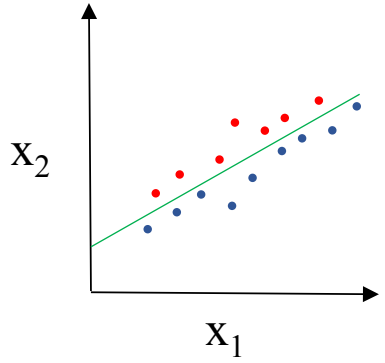
$y \in \{0,1\}$

Data



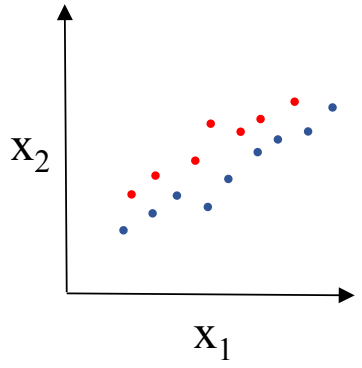
$y = \sigma(\mathbf{w}\mathbf{x})$

Mechanism



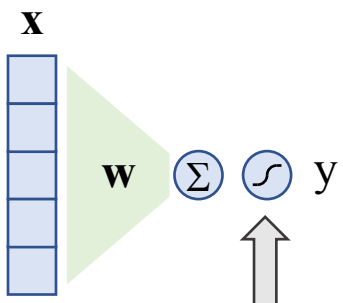
Model

Logistic Regression (1958)

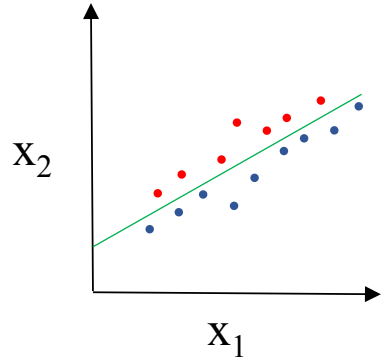


$y \in \{0,1\}$

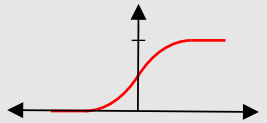
Data



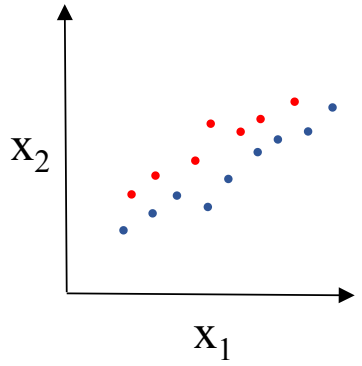
$y = \sigma(\mathbf{w}\mathbf{x})$



Activation Function
A nonlinearity to restrict range of output
Probability of classification:
$\sigma(x) = 1/(1+e^{-x})$

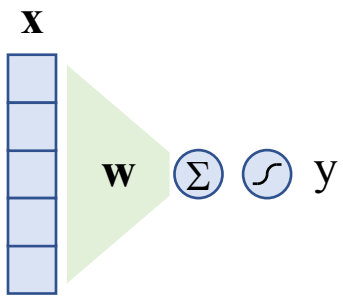


Logistic Regression (1958)



$y \in \{0,1\}$

Data



$y = \sigma(\mathbf{wx})$

Mechanism

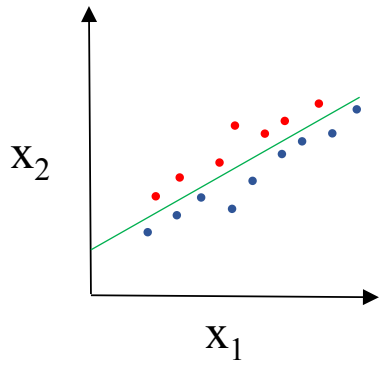
Minimize

$\|\mathbf{y} - \sigma(\mathbf{Xw})\|$

No closed form solution for $\mathbf{w}$

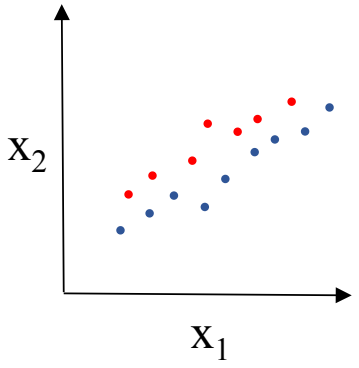
Use gradient descent
(Newton's Method)

Learning



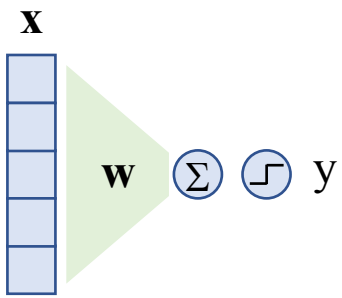
Model

Perceptron (1957)



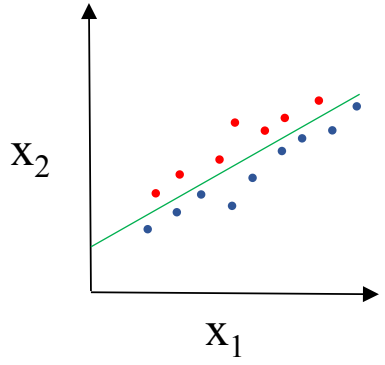
$y \in \{0,1\}$

Data



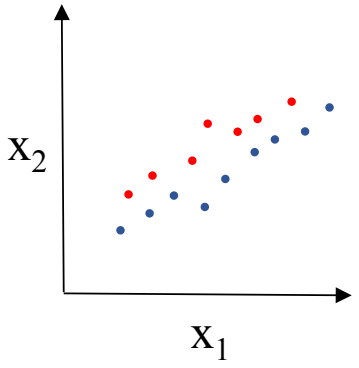
$y = \sigma(\mathbf{w}^T\mathbf{x})$

Mechanism



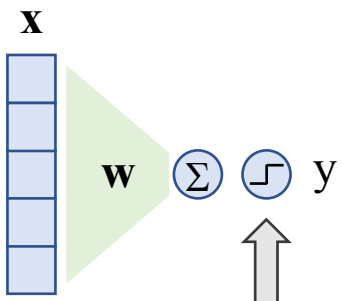
Model

Perceptron (1957)



$y \in \{0,1\}$

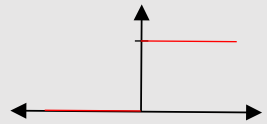
Data



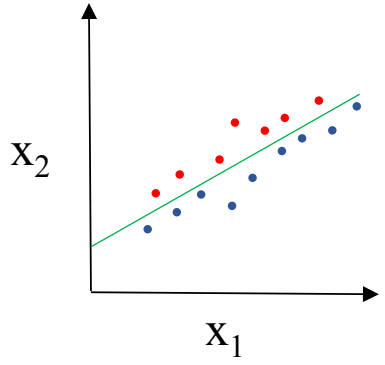
$y = \sigma(\mathbf{w}^T\mathbf{x})$

Activation Function

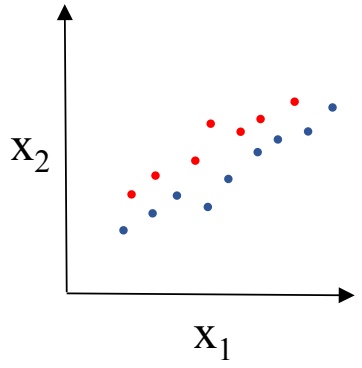
$\sigma(x) = 1$ if $x > 0$, 0 otherwise



$0 < ax_1 + bx_2 + c$

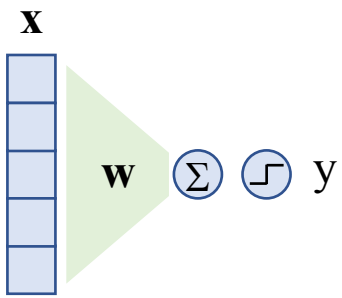


Perceptron (1957)



$y \in \{0,1\}$

Data



$y = \sigma(\mathbf{w}^T\mathbf{x})$

Mechanism

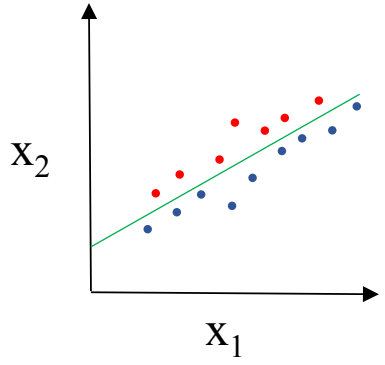
Iterate over input output pairs

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + (y - \sigma(\boldsymbol{w}_t x))\boldsymbol{x}$$

Will converge if separable

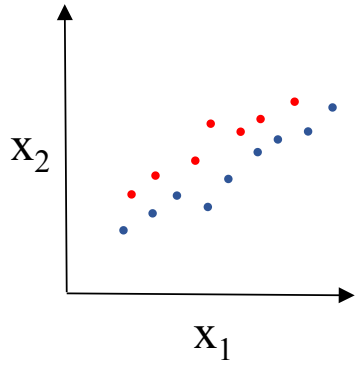
Learning

$0 < ax_1 + bx_2 + c$



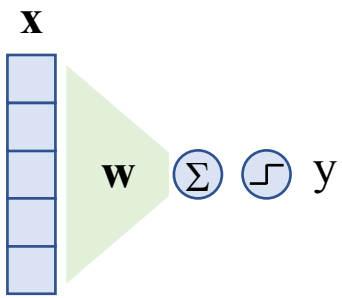
Model

Perceptron (1957)



$y \in \{0,1\}$

Data



$y = \sigma(\mathbf{w}^T\mathbf{x})$

Mechanism

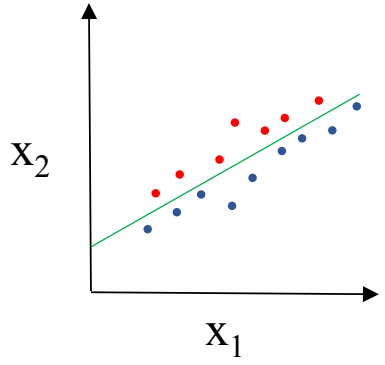
Iterate over input output pairs

$$\boldsymbol{e} = \boldsymbol{y} - \sigma(W_t\boldsymbol{x})$$
$$\Delta W_t = \boldsymbol{e} \otimes \boldsymbol{x}$$

Will converge if separable

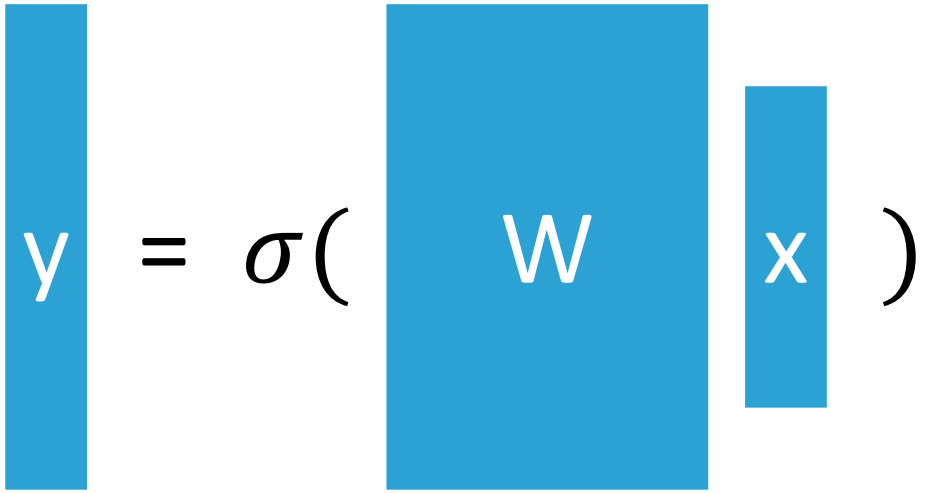
Learning

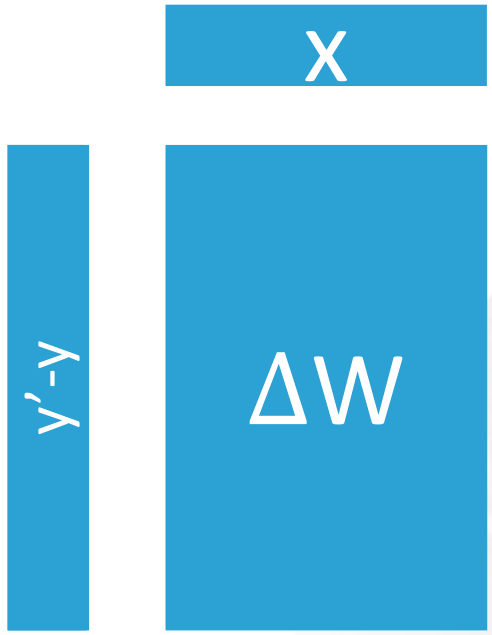
$0 < ax_1 + bx_2 + c$



Model

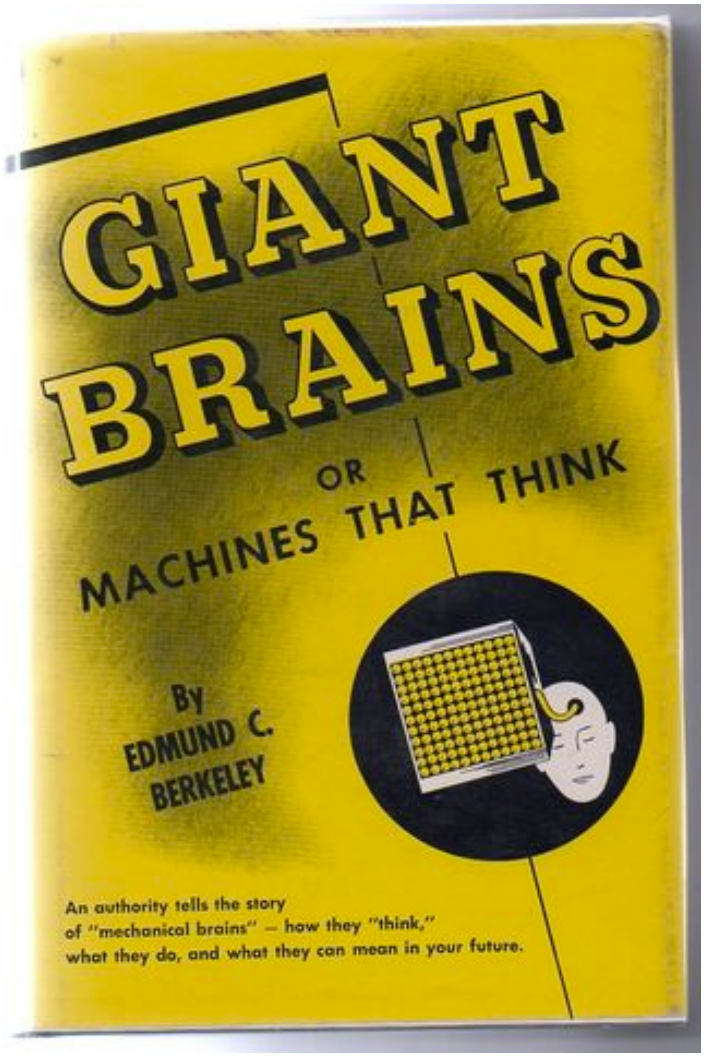
Perceptrons (1957)



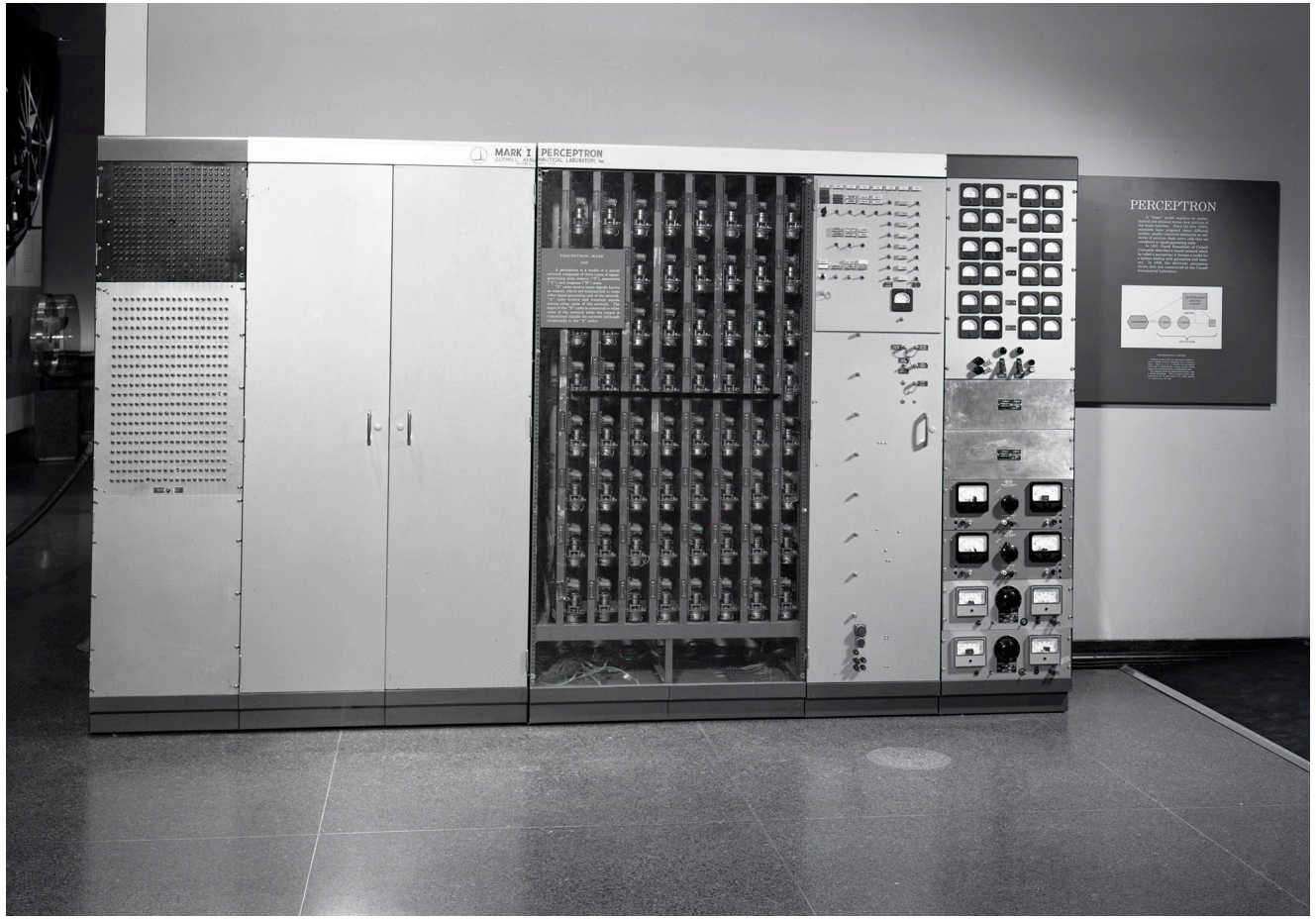


$$\Delta W = e \otimes x$$

Neural Networks Are Great!



Mark 1 Perceptron Hardware

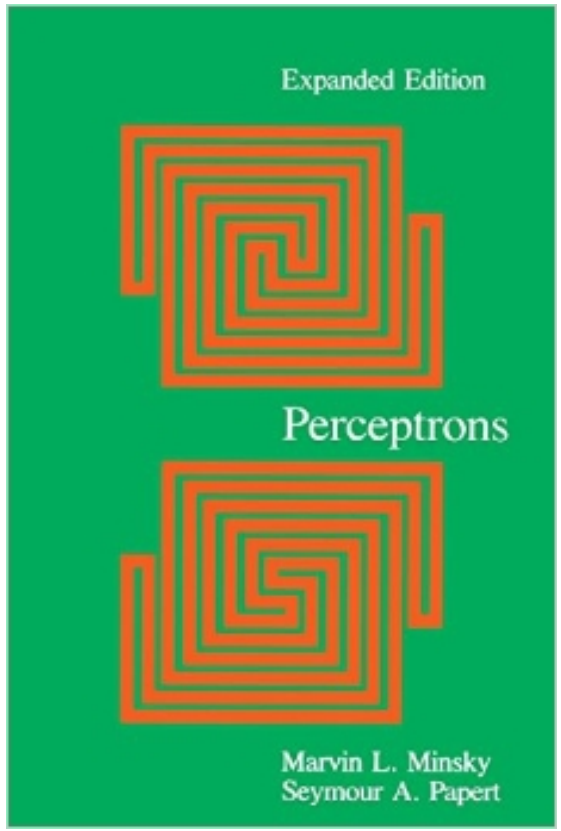


Perceptrons (1960s)

Perceptrons (1960s)

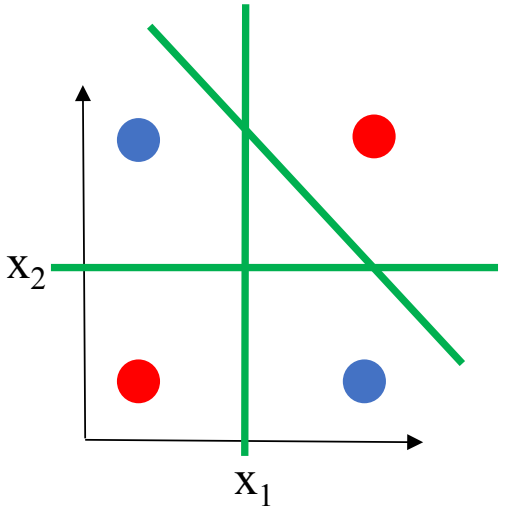
Cloudy Sunny

Twenty Year Intermission



1969

Twenty Year Intermission

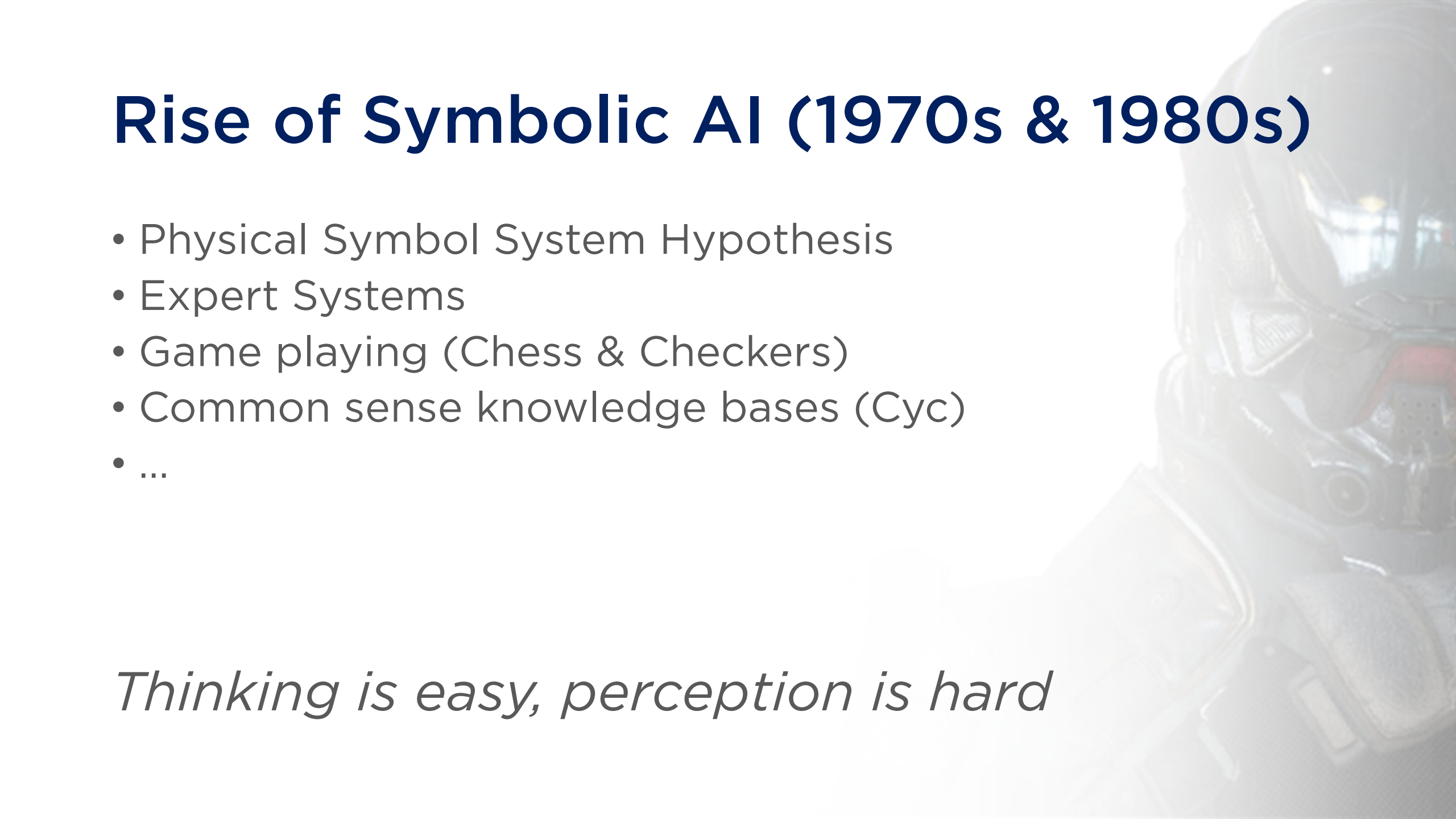


A failure of functionality

Rise of Symbolic AI (1970s & 1980s)

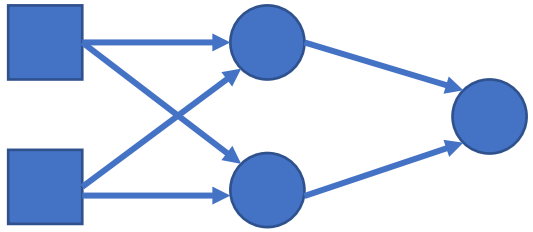
- Physical Symbol System Hypothesis
- Expert Systems
- Game playing (Chess & Checkers)
- Common sense knowledge bases (Cyc)
- …

Thinking is easy, perception is hard



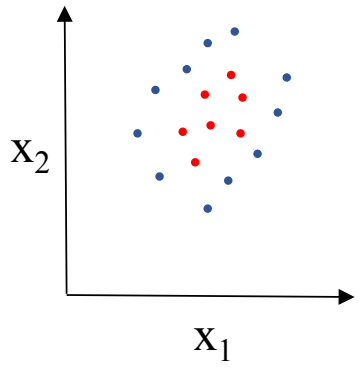
Multilayered Perceptrons

- Two layered networks can classify XOR data set
  - XOR(x,y)=OR(AND(x,y),NOR(x,y))



- No way to learn it

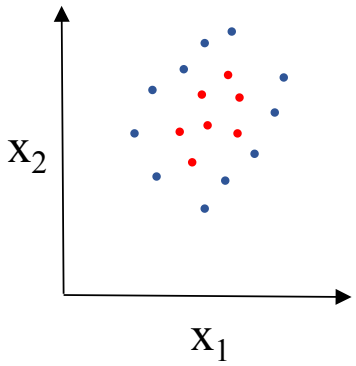
Multilayered Perceptrons (1986)



$y \in \{0,1\}$

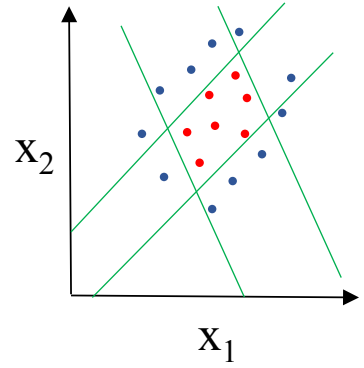
Data

Multilayered Perceptrons (1986)



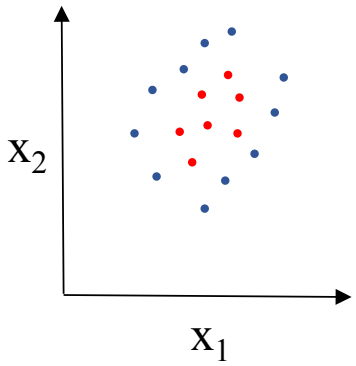
$y \in \{0,1\}$

Data



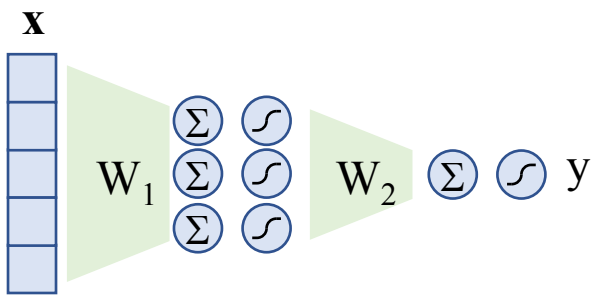
Model

Multilayered Perceptrons (1986)

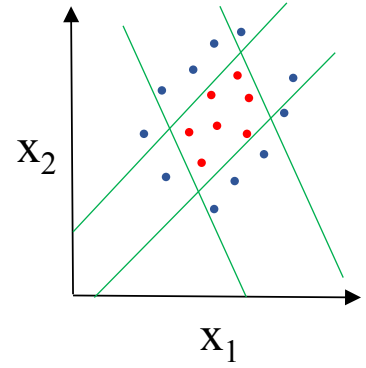


$y \in \{0,1\}$

Data

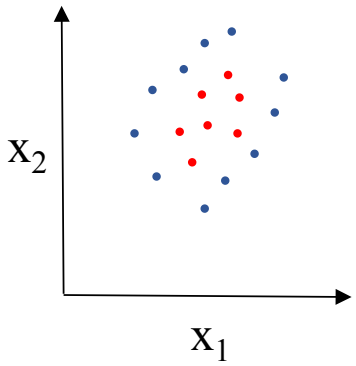


$y = f_2(f_1(\mathbf{x}, W_1), W_2)$



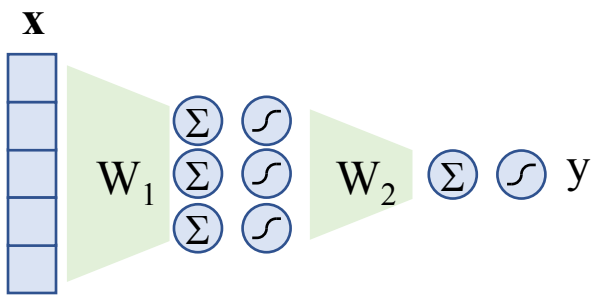
Model

Multilayered Perceptrons (1986)



$y \in \{0,1\}$

Data



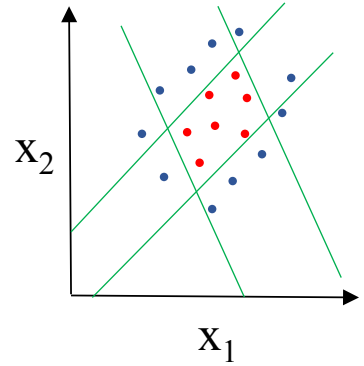
$$y = f_2(f_1(\mathbf{x},W_1),W_2)$$

Minimize

$$\|\mathbf{y} - f_2(f_1(\mathbf{x},W_1),W_2)\|$$

No closed form solution for $\mathbf{w}$

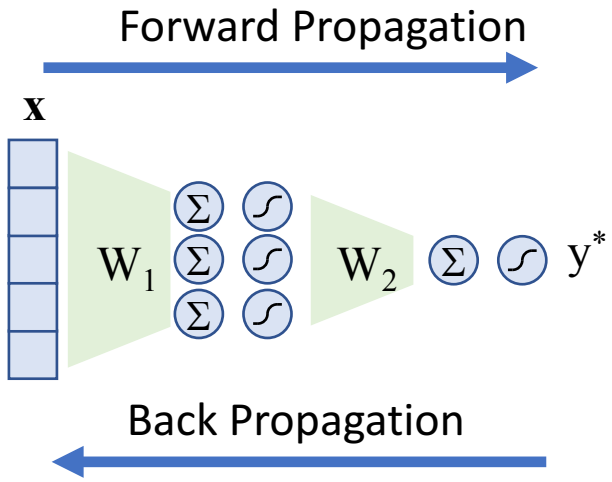
Minimize with gradient descent (back propagation)



Model

Back Propagation of Error

- Push error into nested functions
- Error metric
  - $E = ||y - f_2(f_1(x,W_1),W_2)||$
- Continuous & differentiable
- Gradient descent on error with respect to weights

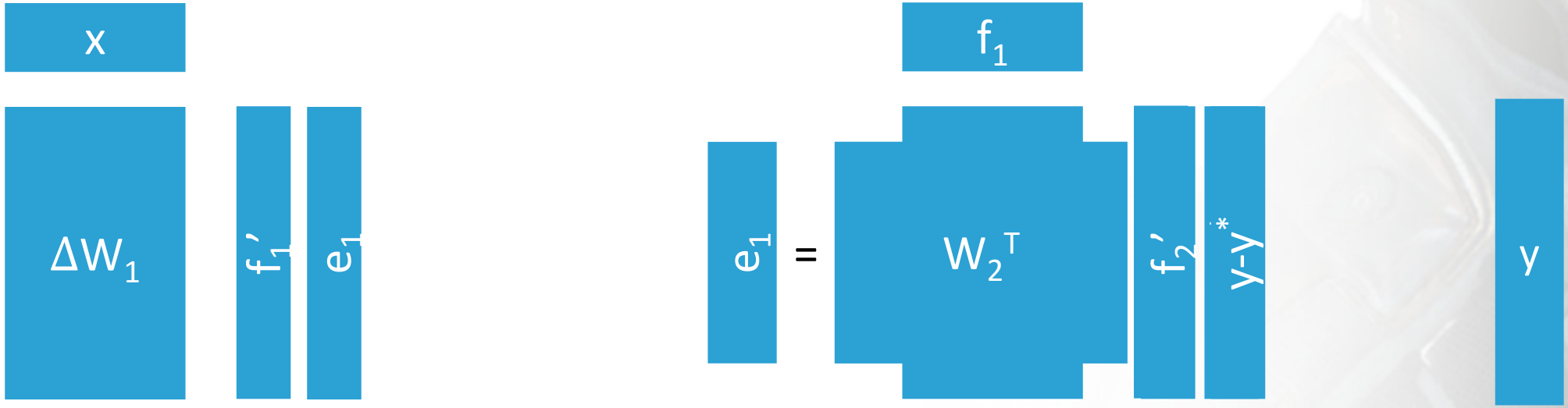


$$y^* = f_2(f_1(\mathbf{x},W_1),W_2)$$

Back Propagation of Error

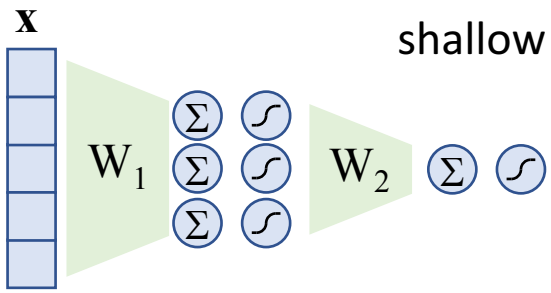
$$\sigma(W_1 \; x) = f_1$$

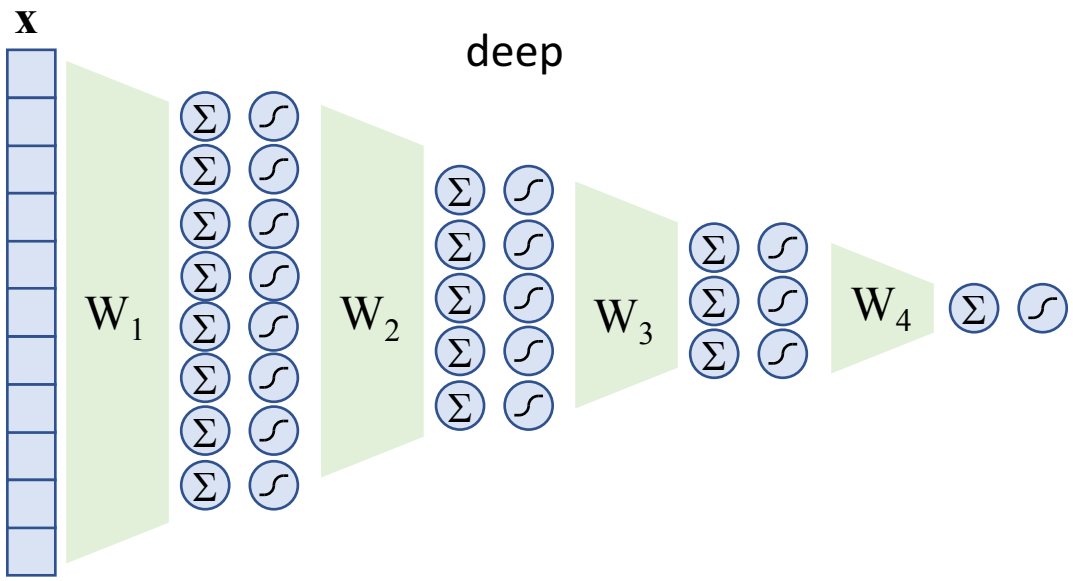
$$\sigma(W_2 \; f_1) = y^*$$



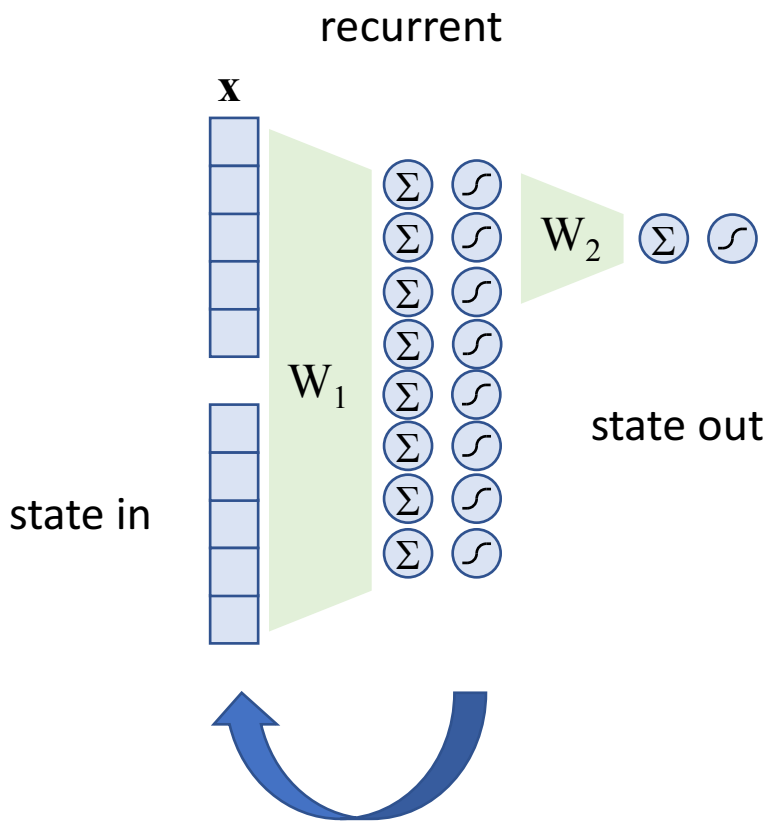
Neural Networks Are Great! (Again)

Deep Networks





Recurrent Networks



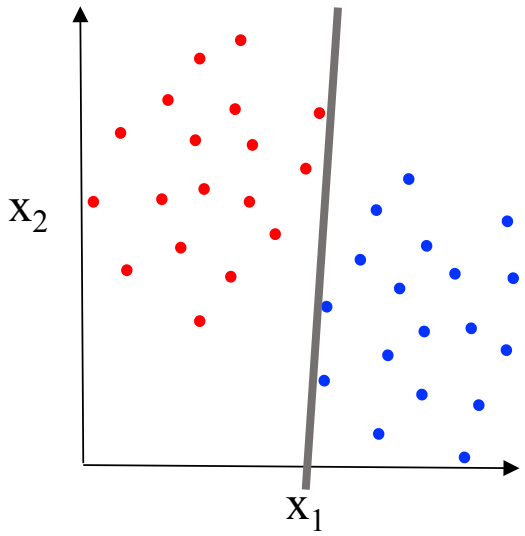
Three Problems

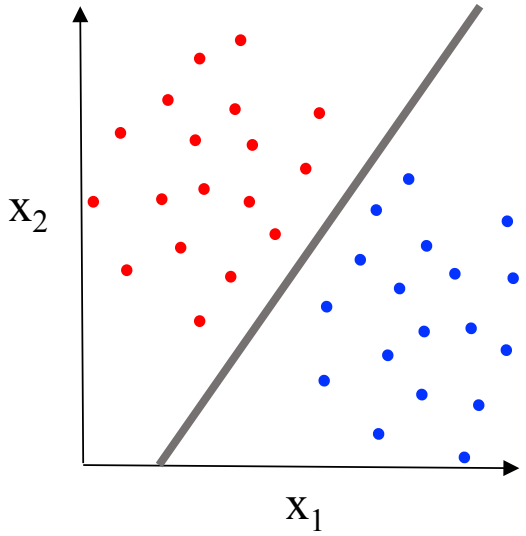
- Vanishing gradients in deep and recurrent networks
- High dimensional parameter spaces
- Long training times

Support Vector Machines (1992)

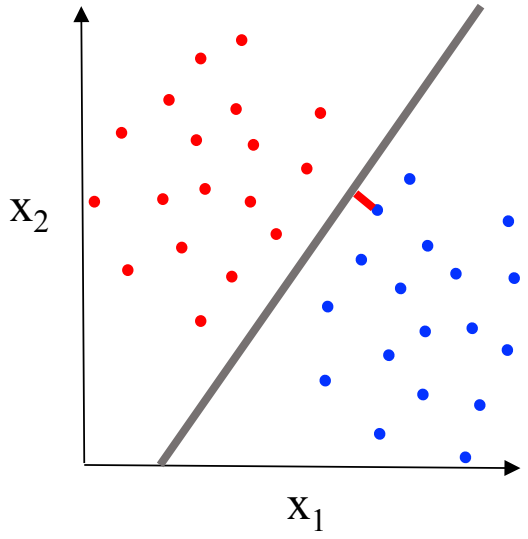
- Margin optimization (1963)
- Kernel Trick

Support Vector Machines (1992)





Support Vector Machines (1992)



Maximize Margin

Learning SVM
Write down constraints
Solve Quadratic Program

Support Vector Machines (1992)

- Learns linear mapping
- Non-linear mappings handled by “Kernel Trick”
  - Parabolic boundary
  - [x,y] -> [x,y,x^2,y^2]

Neural Networks on Spark

Linear Regression

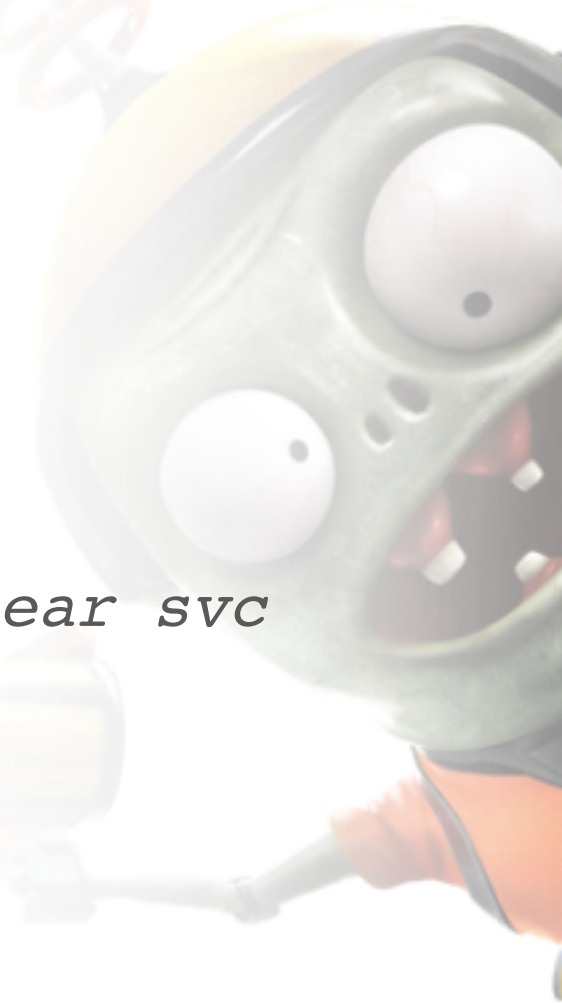
```
val lr = new LinearRegression().
 setMaxIter(10).
 setRegParam(0.3)

// Fit the model
val lrModel = lr.fit(training)


// Print the coefficients and intercept for logistic
regression
val c = lrModel.coefficients
val i = lrModel.intercept
println(s"Coefficients: ${c} Intercept: ${i}")
```

Logistic Regression

```
val lr = new LogisticRegression().
  setMaxIter(10).
  setRegParam(0.3)

// Fit the model
val lrModel = lr.fit(training)

// Print the coefficients and intercept for logistic
regression
```

Support Vector Machine

```
val lsvc = new LinearSVC().
  setMaxIter(10).
  setRegParam(0.1)

// Fit the model
val lsvcModel = lsvc.fit(training)

// Print the coefficients and intercept for linear svc
val c = lsvcModel.coefficients
val i = lsvcModel.intercept
println(s"Coefficients: ${c} Intercept: ${i}")
```



Multilayered Perceptron

```
// specify layers for the neural network:
// input layer of size 4 (features), two intermediate of size 5 and 4
// and output of size 3 (classes)
val layers = Array[Int](4, 5, 4, 3)

// create the trainer and set its parameters
val trainer = new MultilayerPerceptronClassifier().
  setLayers(layers).
  setBlockSize(128).
  setSeed(1234L).
  setMaxIter(100)

// train the model
val model = trainer.fit(train)

// compute accuracy on the test set
val result = model.transform(test)
val predictionAndLabels = result.select("prediction", "label")
val evaluator = new MulticlassClassificationEvaluator().
  setMetricName("accuracy")

println("Test set accuracy = " +
        evaluator.evaluate(predictionAndLabels))
```