Introduction to Neural Computation

Prof. Michale Fee
MIT BCS 9.40 — 2018

Lecture 15
Perceptrons and Matrix Operations

Learning Objectives for Lecture 15

- Perceptrons and perceptron learning rule
- Neuronal logic, linear separability, and invariance
- Two-layer feedforward networks
- Matrix algebra review
- Matrix transformations

Review

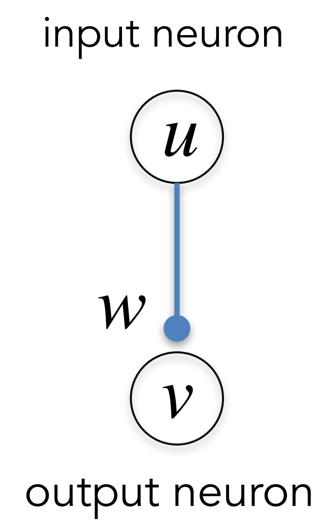
- We have been considering neural networks that use firing rates, rather than spike trains. ('rate model')

- Synaptic input is the firing rate of the input neuron times a synaptic weight w.

$$I_s = wu$$

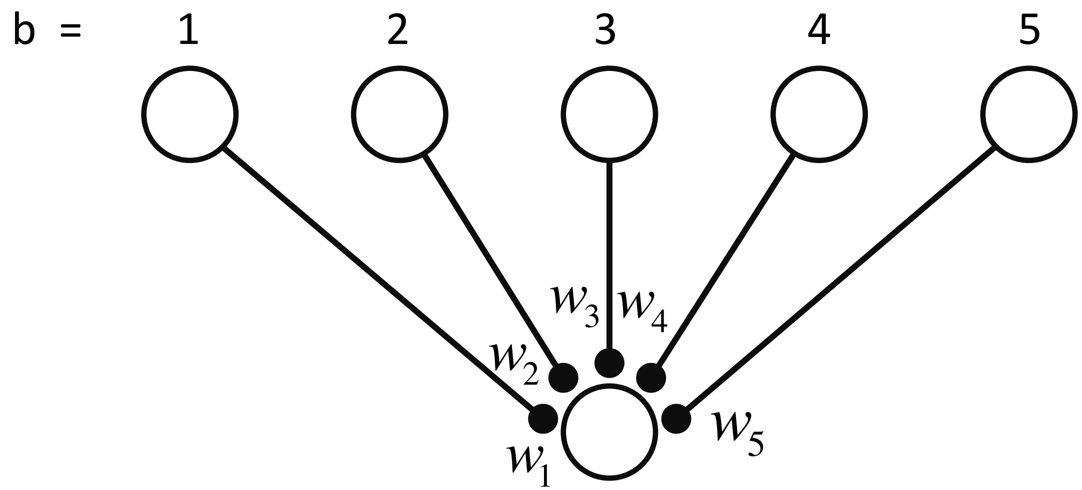
- The output firing rate is some non-linear function of the synaptic input.

$$v = F[I_s] = F[wu]$$



Review

- We generalized this model to the case when there are many synaptic inputs…



v = output firing rate

Input firing rates

$$\left[u_1 , u_2 , u_3 , \ldots\ u_{n_b} \right] = \vec{u}$$

Input synaptic weights

$$\left[w_1 , w_2 , w_3 , \ldots\ w_{n_b} \right] = \vec{w}$$

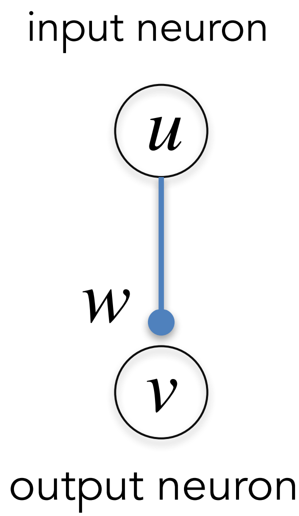
$$I_s = w_1 u_1 + w_2 u_2 + w_3 u_3 + \ldots = \sum_b w_b u_b = \vec{w} \cdot \vec{u}$$

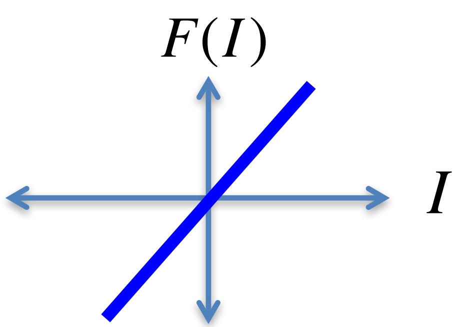
$$\boxed{v = F\left[\vec{w} \cdot \vec{u}\right]}$$

Review

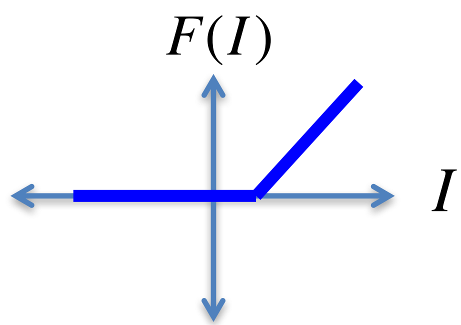
- The output firing rate is some non-linear function of the synaptic input.

$$v = F[I_s] = F[wu]$$

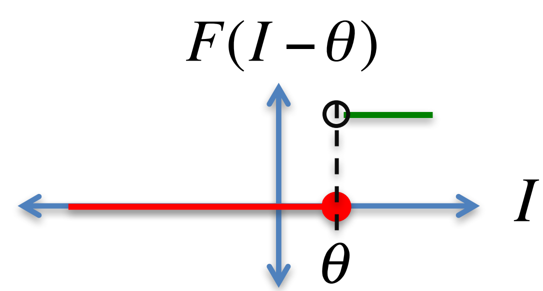




linear neuron



Linear threshold neuron

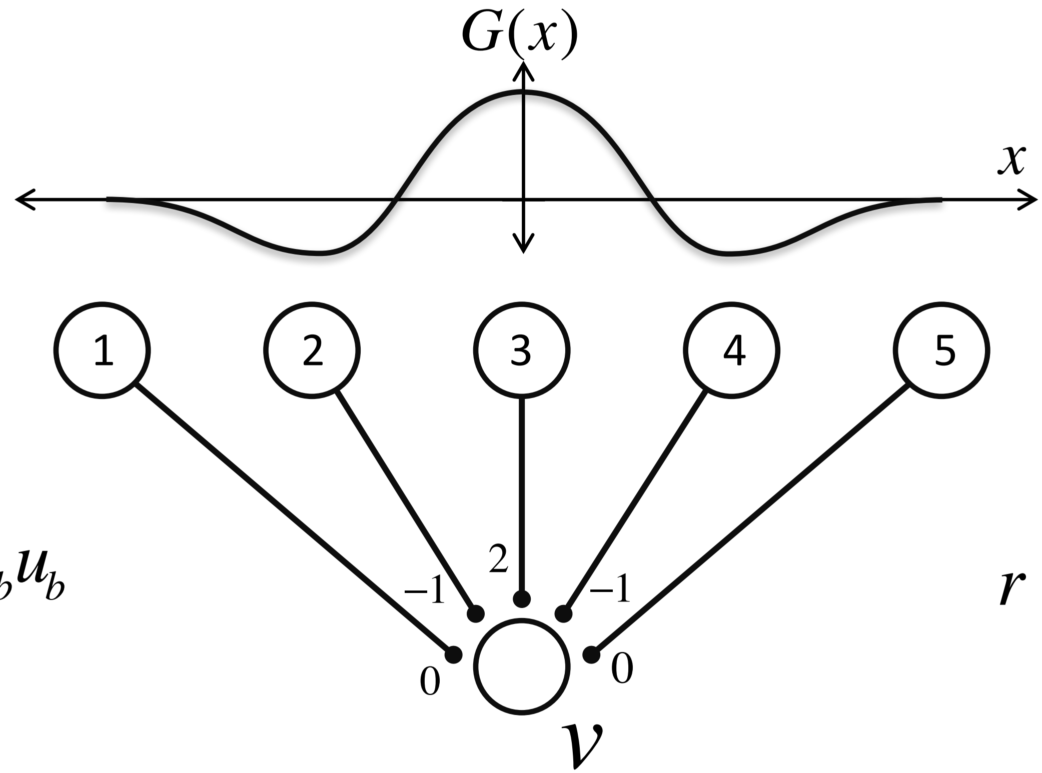


Binary threshold neuron

How to build a receptive field

- We can see that the choice of weights allows us to specify the receptive field of our output neuron

$$v = \sum_b w_b u_b$$



$$r = \int G(x) I(x)$$

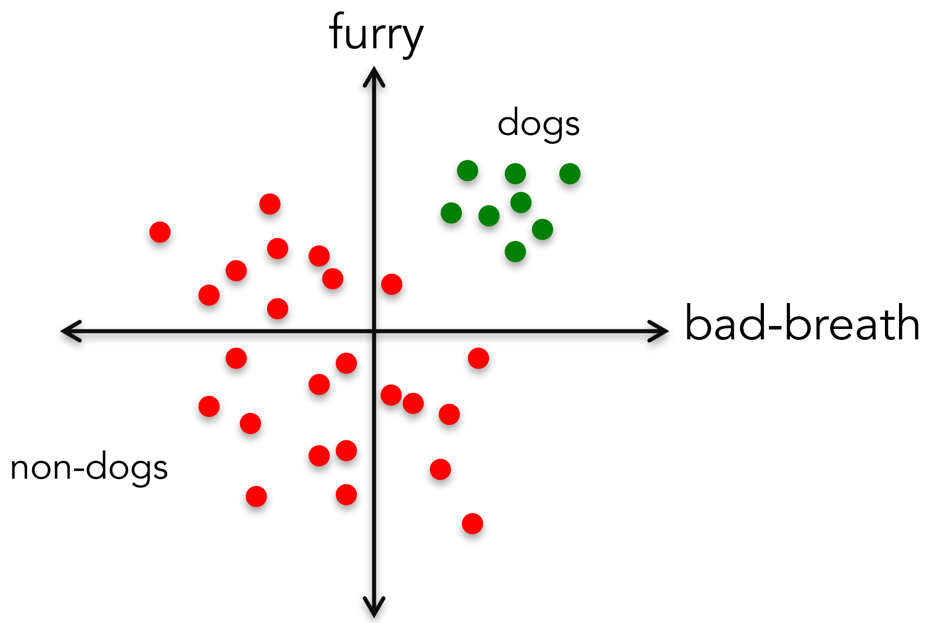
$$\vec{w} = \left[0\ ,\ -1\ ,\ 2\ ,\ -1\ ,\ 0\right]$$

Learning Objectives for Lecture 15

- Perceptrons and perceptron learning rule
- Neuronal logic, linear separability, and invariance
- Two-layer feedforward networks
- Matrix algebra review
- Matrix transformations

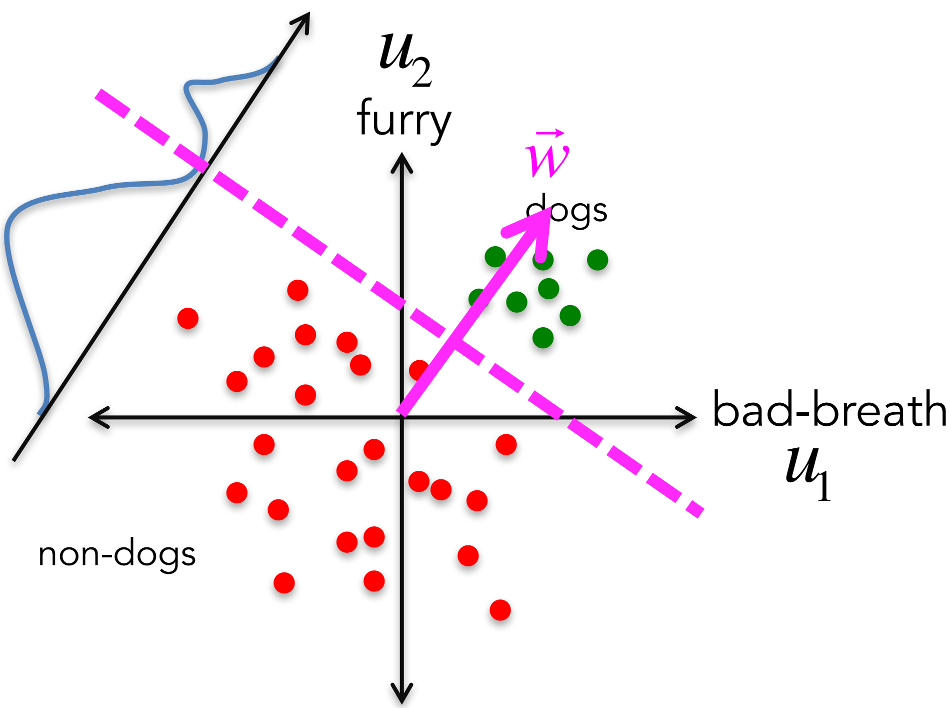
Perceptrons

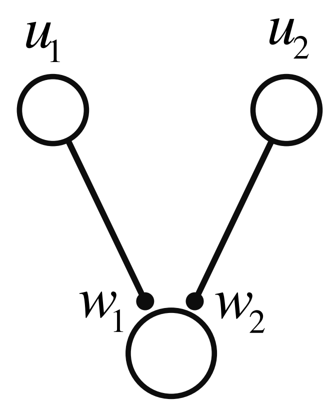
- A perceptron carries out classification of inputs that represent features.



Perceptrons

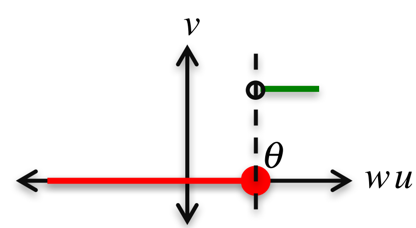
- A perceptron carries out classification of inputs that represent features.





$I_s = \vec{w} \cdot \vec{u}$

Binary Threshold Neuron for decision-making



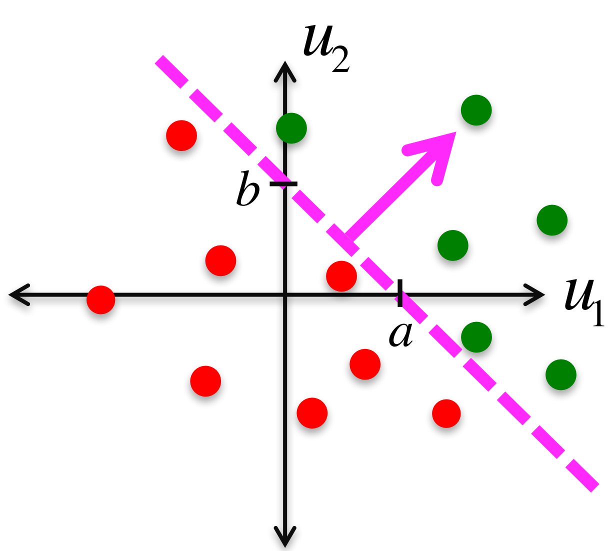
$v = F(\vec{w} \cdot \vec{u} - \theta)$

Classification in two dimensions

$$v = F(\vec{w} \cdot \vec{u} - \theta)$$

The decision boundary is $\vec{w} \cdot \vec{u} = \theta$

- Let's calculate the weight vector $\vec{w} = (w_1 \, , w_2)$ that gives us the decision boundary shown below. Assume $\theta = 1$.



We have two points on the decision boundary we know, and two unknowns…

$$\vec{u}_a = (a \, , 0) \qquad \vec{u}_a \cdot \vec{w} = \theta$$

$$\vec{u}_b = (0 \, , b) \qquad \vec{u}_b \cdot \vec{w} = \theta$$

$$\vec{w} = (1/a \, , 1/b)$$

- This is easy to do (by eye!) in two dimensions – but how about in higher dimensions?

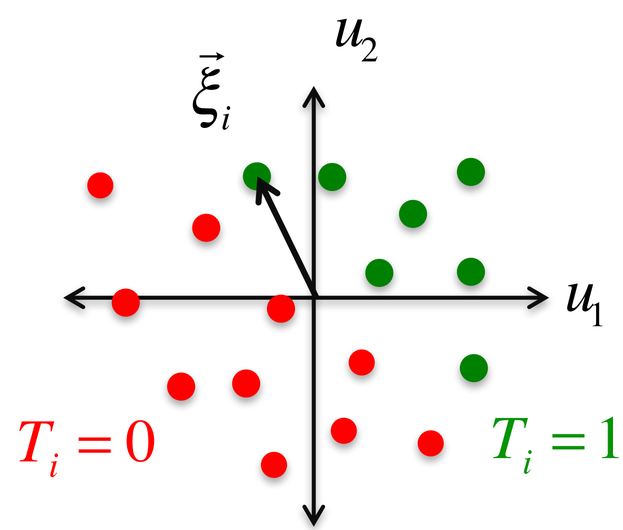
Classification in higher dimensions

- Let's say we have n observations of our inputs

$$\vec{u} = \vec{\xi}_i \quad , \quad i = 1,2, \ldots n$$

- After each observation, we are told whether this input corresponds to a dog.

$$T_i = \begin{cases} 1 & for\ yes \\ 0 & for\ no \end{cases} \quad , \quad i = 1,2, \ldots n$$



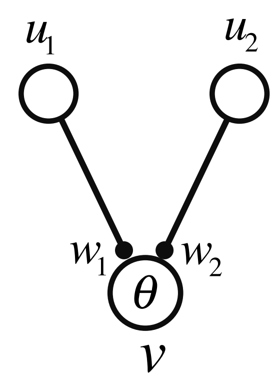
- We want to find $\vec{w}$, such that

$$v_i = \mathrm{step}\left(\vec{w} \cdot \vec{\xi}_i - \theta\right) = T_i \quad , \quad for\ all\ i$$

Perceptron learning

- How would we find the weight vector w that separates dogs from non-dogs?

- Each observation $\vec{u}_i\ ,T_i$ gives us information we can use to construct $\vec{w}$. This is called supervised learning.



- We can learn w iteratively: i.e., on each observation we will update our estimate of $\vec{w}$

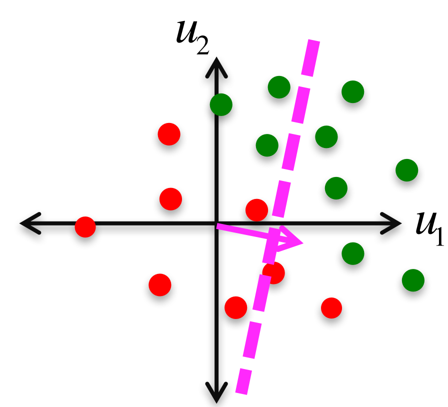
$$\vec{w} \rightarrow \vec{w} + \Delta\vec{w}$$

Rosenblatt, 1957

- How do we start?

  - we can start with a random set of weights

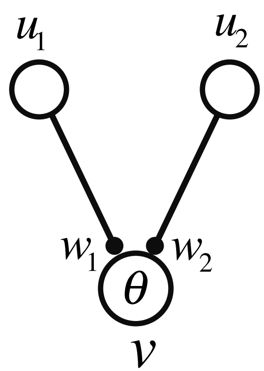
  - or start with zero weights $\vec{w} = 0$



Perceptron learning rule

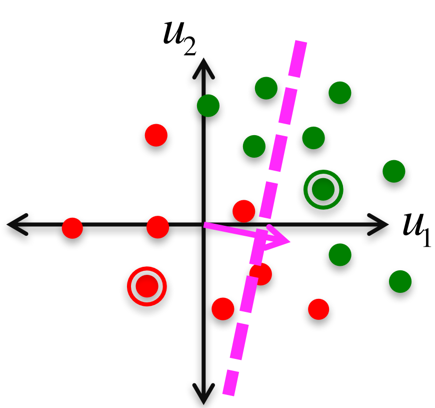
- On each observation of $\vec{u} = \vec{\xi}_i$, we use our current estimate of $\vec{w}$ to classify $\vec{\xi}_i$:

$$v_i = \mathrm{step}\left(\vec{w} \cdot \vec{\xi}_i - \theta\right)$$



- Compare our classification to the right answer…

○ If $v_i = T_i$ then we were right !
so don't do anything: $\Delta\vec{w} = 0$



Perceptron learning rule

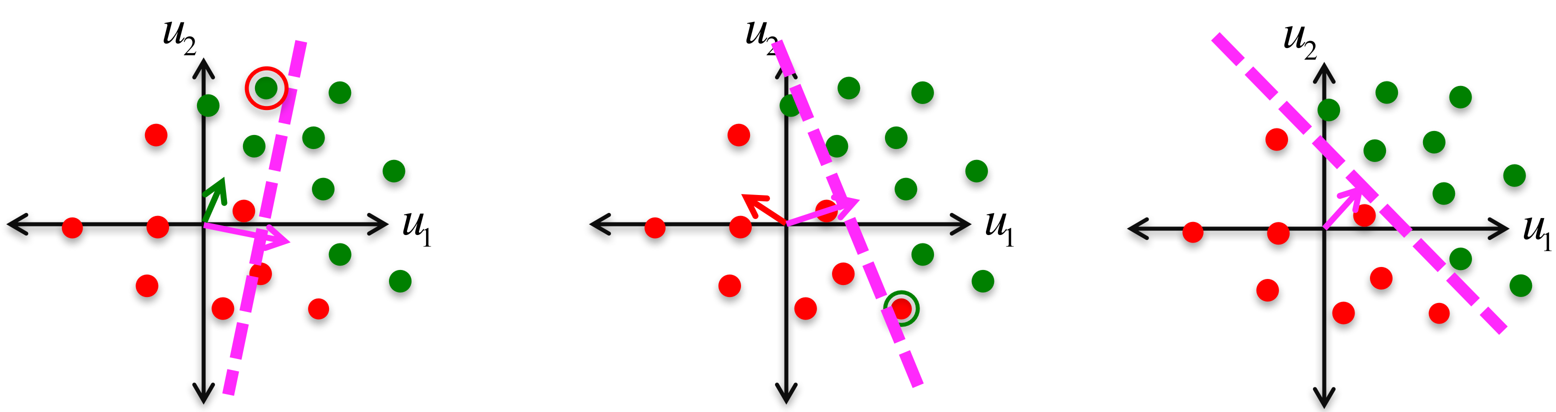
- If $v_i \neq T_i$ then we were wrong, so update $\vec{w}$.

$$\Delta \vec{w} = \begin{cases} \eta \vec{\xi}_i \,, & if\ T = 1 \\ -\eta \vec{\xi}_i \,, & if\ T = 0 \end{cases}$$

η is the 'learning rate'

Increase w in the direction of $\vec{\xi}_i$ if the correct answer was 1,

away from $\vec{\xi}_i$ if the correct answer was 0.



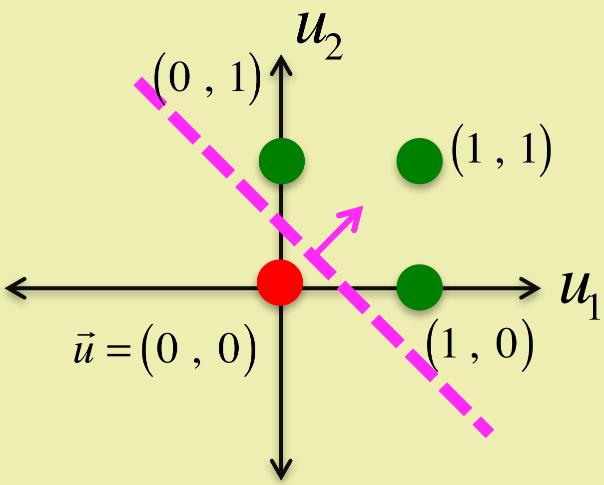
Learning Objectives for Lecture 15

- Perceptrons and perceptron learning rule
- Neuronal logic, linear separability, and invariance
- Two-layer feedforward networks
- Matrix algebra review
- Matrix transformations

Neuronal Logic

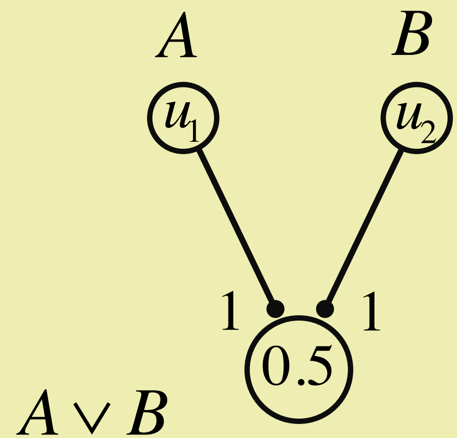
- The perceptron naturally implements simple logic gates…

OR

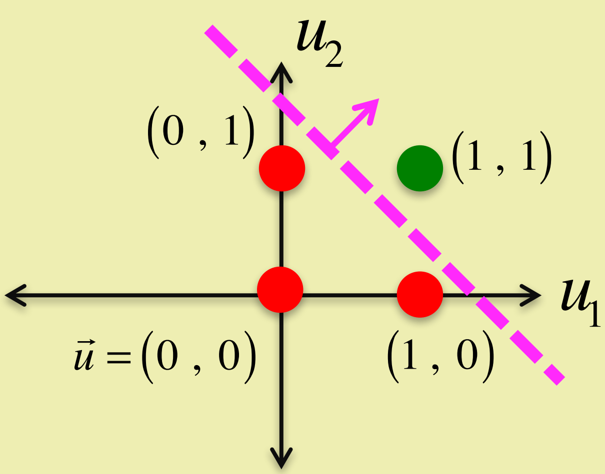


$$\vec{w} = (1 , 1)$$

$$\theta = 0.5$$

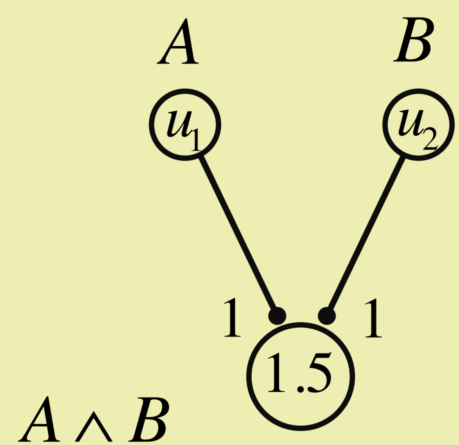


AND



$$\vec{w} = (1 , 1)$$

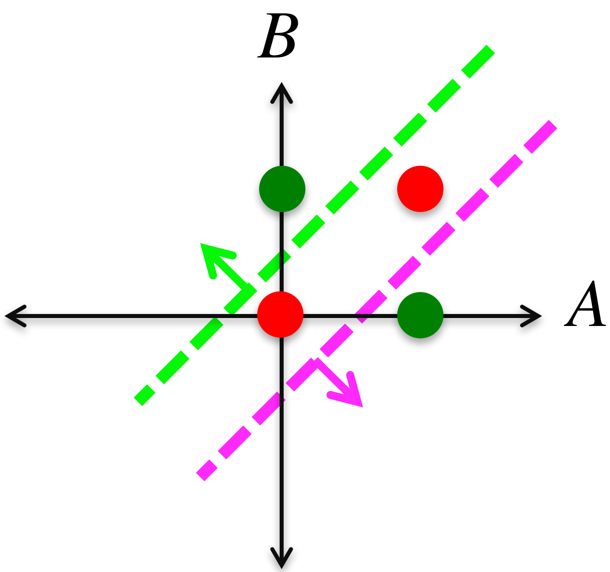
$$\theta = 1.5$$



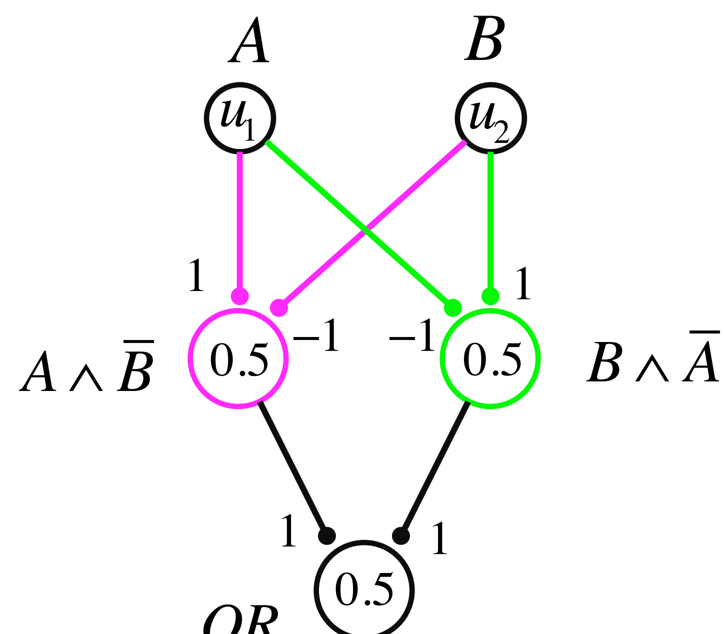
Linear separability

- There are some classification problems the perceptron cannot solve.

- Exclusive OR (XOR) – A or B but not both



Multi-layer perceptron

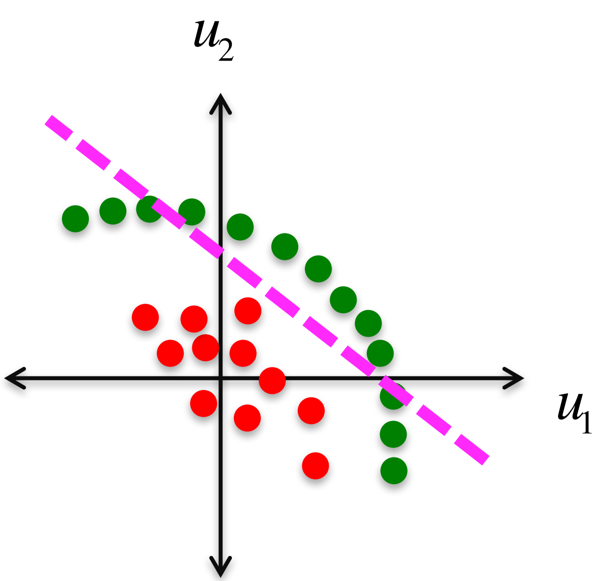


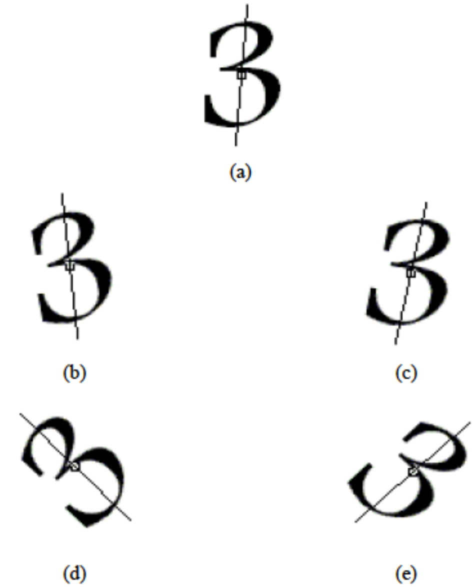
- The problem of linear separability

Linear separability

- Classification problems are difficult because of transformations such as translation, rotation, scale
- In high dimensional space, images that are related by invariant transformations can be thought of as existing on 'manifolds'

Usually not linearly separable

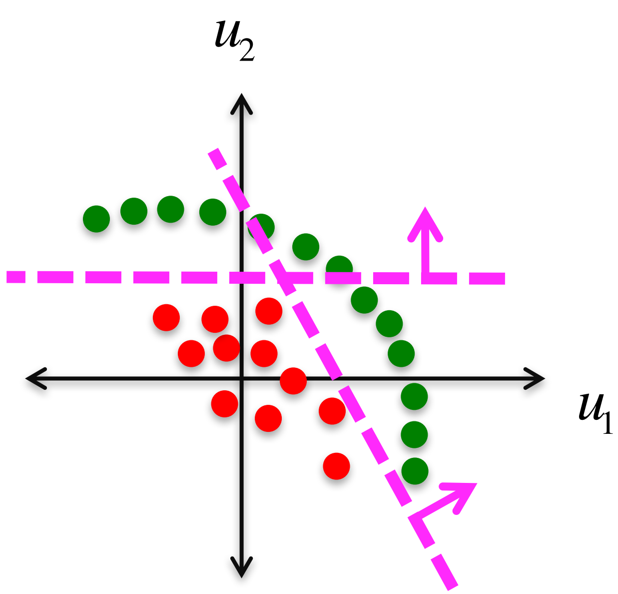


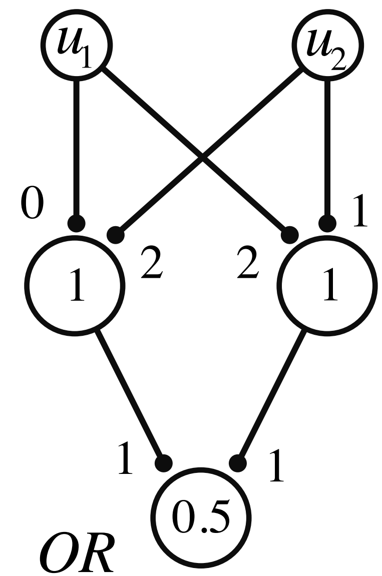


Invariance

- Multilayer perceptrons can sometimes solve the problem!
- We can break the classification into several linearly separable problems

Multi-layer perceptron





Deep neural networks

- Multilayer perceptrons can sometimes solve the problem!

DiCarlo and Yamins, 2015

Learning Objectives for Lecture 15

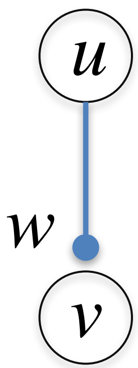
- Perceptrons and perceptron learning rule
- Neuronal logic, linear separability, and invariance
- Two-layer feedforward networks
- Matrix algebra review
- Matrix transformations

More complex networks

- We have considered increasingly complex network models

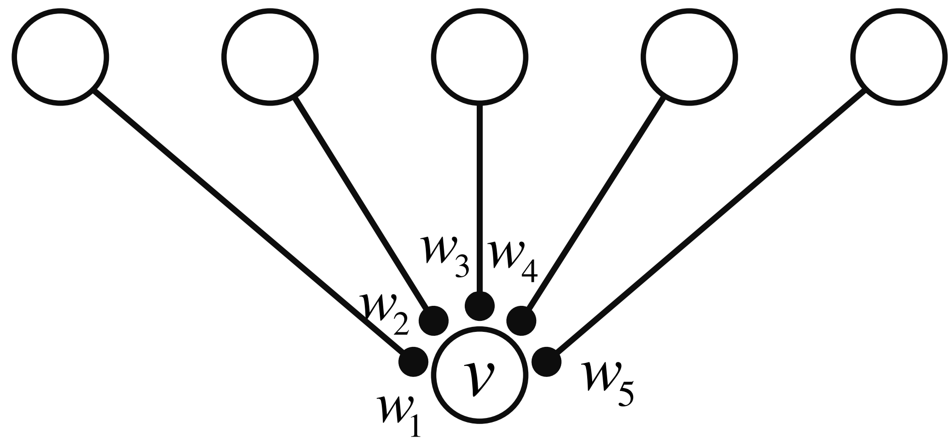
input firing rates $\left[u_1\,, u_2\,, u_3\,, \ldots\ u_{n_b} \right] = \vec{u}$

input neuron



output neuron

b = 1 2 3 4 5



$$I_s = wu$$

$$v = F\left[wu\right]$$

$$I_s = \sum_b w_b u_b \quad = \vec{w} \cdot \vec{u}$$

$$v = F\left[\vec{w} \cdot \vec{u}\right]$$

Two-layer feed-forward network

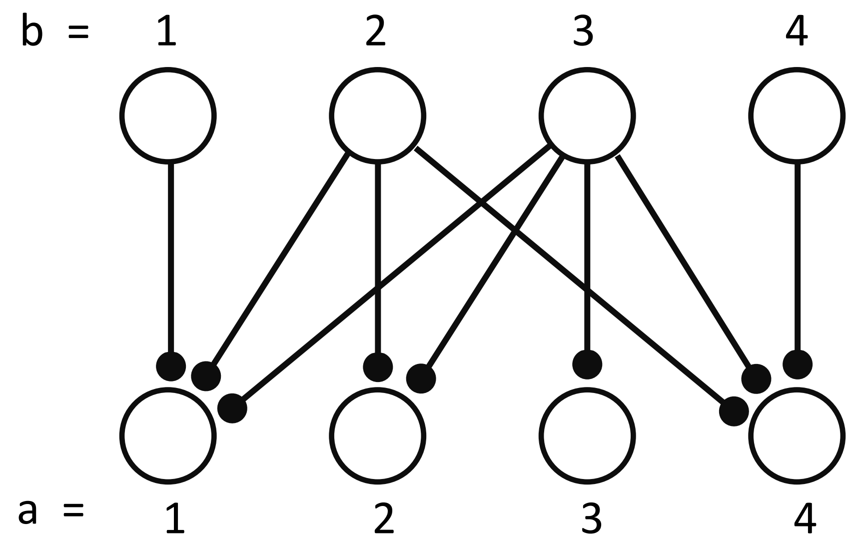
- We can expand our set of output neurons to make a more general network…

input firing rates

$$\left[u_1 , u_2 , u_3 ,\dots\ u_{n_b} \right] = \vec{u}$$

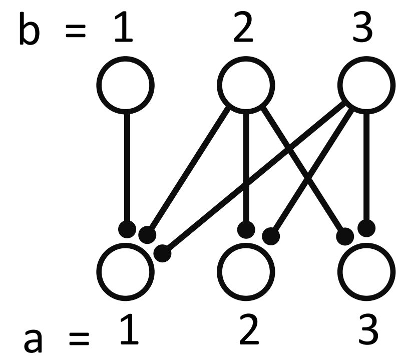
output firing rates

$$\left[v_1 , v_2 , v_3 ,\dots\ v_{n_a} \right] = \vec{v}$$



Two-layer feed-forward network

- We can write down the firing rates of our output neurons as follows:



$$v_1 = \vec{w}_{a=1} \cdot \vec{u} \qquad v_1 = \sum_b W_{1b} u_b$$

$$v_2 = \vec{w}_{a=2} \cdot \vec{u} \qquad v_2 = \sum_b W_{2b} u_b$$

$$v_3 = \vec{w}_{a=3} \cdot \vec{u} \qquad v_3 = \sum_b W_{3b} u_b$$

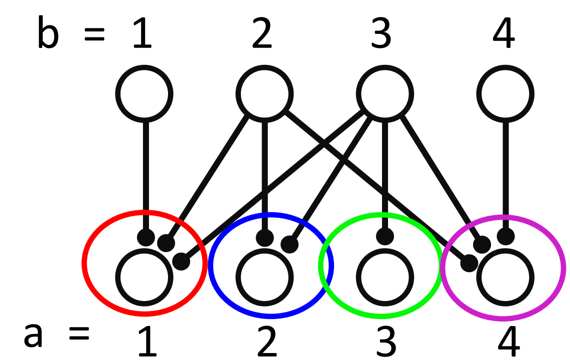
$$\vec{v} = W \, \vec{u}$$

$$v_a = \vec{w}_a \cdot \vec{u} \qquad v_a = \sum_b W_{ab} u_b$$

- Our feed-forward network implements matrix multiplication!

Two-layer feed-forward network

- We have a weight from each of our input neurons onto each of our output neurons.



- We write the weights as a matrix.

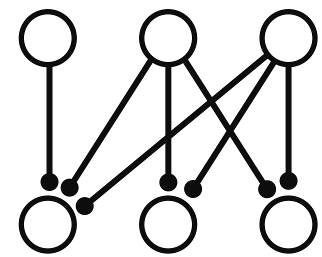
weight matrix

$$
W_{ab} = \begin{array}{c|c} & b = \; 1 \quad\; 2 \quad\; 3 \quad\; 4 \\ \begin{array}{c} a = \\ 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{array}{c} \\ \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \end{bmatrix} \end{array} \end{array} = \begin{bmatrix} \vec{w}_{a=1} \\ \vec{w}_{a=2} \\ \vec{w}_{a=3} \\ \vec{w}_{a=4} \end{bmatrix}
$$

Two-layer feed-forward network

- We can write down the firing rates of our output neurons as a matrix multiplication.

$$\vec{v} = W\,\vec{u} \qquad v_a = \sum_b W_{ab} u_b$$



$$\vec{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} \vec{w}_{a=1} \cdot \vec{u} \\ \vec{w}_{a=2} \cdot \vec{u} \\ \vec{w}_{a=3} \cdot \vec{u} \end{bmatrix}$$

- Dot product interpretation of matrix multiplication

Learning Objectives for Lecture 15

- Perceptrons and perceptron learning rule
- Neuronal logic, linear separability, and invariance
- Two-layer feedforward networks
- Matrix algebra review
- Matrix transformations

Matrix algebra

- Vectors are collections of numbers.

Let's say we make measurements of two different things, x_a and x_b , at a particular time.

$$\vec{x} = \begin{pmatrix} x_a \\ x_b \end{pmatrix}$$

- Matrices are collections of vectors

Now we measure x_a and x_b at three different times

$$\vec{x}_1 = \begin{pmatrix} 1 \\ 3 \end{pmatrix} \quad \vec{x}_2 = \begin{pmatrix} -2 \\ 4 \end{pmatrix} \quad \vec{x}_3 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

We can write all our measurements down as a matrix

$$X = \left(\vec{x}_1 \middle| \vec{x}_2 \middle| \vec{x}_3 \right) = \begin{pmatrix} 1 & -2 & 0 \\ 3 & 4 & 1 \end{pmatrix}$$

```
x1=[1;3]       %column
x2=[-2;4]      %column
x3=[0;1]       %column
X=[x1,x2,x3]   %concatenate
```

Matrix algebra

- Labeling matrix elements

$$X = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \end{pmatrix}$$ 2 rows x 3 columns

- Matrix transpose flips the rows and columns

$$X^T = \begin{pmatrix} x_{11} & x_{21} \\ x_{12} & x_{22} \\ x_{13} & x_{23} \end{pmatrix}$$ 3 rows x 2 columns

- Symmetric matrix

$$X = \begin{pmatrix} a & c \\ c & b \end{pmatrix} \qquad X^T = X$$

Matrix multiplication

- In general, we carry out matrix multiplication by taking dot products of all the rows of the first matrix with all the columns of the second matrix.

$$A = \begin{pmatrix} 1 & -2 & 0 \\ 3 & 4 & 1 \end{pmatrix} \qquad B = \begin{pmatrix} 4 & 2 \\ 7 & 3 \\ -1 & 0 \end{pmatrix}$$

$$AB = \underset{\text{m x k}}{\begin{pmatrix} 1 & -2 & 0 \\ 3 & 4 & 1 \end{pmatrix}} \underset{\text{k x n}}{\begin{pmatrix} 4 & 2 \\ 7 & 3 \\ -1 & 0 \end{pmatrix}} = \underset{\text{m x n}}{\begin{pmatrix} -10 & -4 \\ 39 & 18 \end{pmatrix}}$$

$$\begin{pmatrix} 4-14+0 & 2-6+0 \\ 12+28-1 & 6+12+0 \end{pmatrix}$$

$$AB \neq BA$$

Matrix algebra

- We can still do all our vector operations on the vectors in X
- For example, let's take the dot product of each of our vectors $\vec{x}_1, \vec{x}_2, \vec{x}_3$ with another vector $\vec{v}$.

$$\vec{v} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

- We can do this in two different ways:

$$\vec{y} = \vec{v}^T X = \begin{pmatrix} 1 & -1 \end{pmatrix} \begin{pmatrix} 1 & -2 & 0 \\ 3 & 4 & 1 \end{pmatrix} = \begin{pmatrix} -2 & -6 & -1 \end{pmatrix}$$

1 x 2   2 x 3   1 x 3

Matrix algebra

- Alternatively, let's take the dot product of each of our vectors $\vec{x}_1, \vec{x}_2, \vec{x}_3$ with another vector $\vec{v}$.

- We can do it like this…

$$\vec{y} = X^T \vec{v} = \underset{3 \times 2}{\begin{pmatrix} 1 & 3 \\ -2 & 4 \\ 0 & 1 \end{pmatrix}} \underset{2 \times 1}{\begin{pmatrix} 1 \\ -1 \end{pmatrix}} = \underset{3 \times 1}{\begin{pmatrix} -2 \\ -6 \\ -1 \end{pmatrix}}$$

```
v=[1;-1]  %column
y=X'*v    %' is transpose
```

- Note that matrix multiplication takes the dot product of each of the rows of the first matrix with each of the columns of the second matrix!

Identity matrix

- When multiplying numbers, the number 1 has a special property:

$$a \cdot 1 = a$$

- Is there a matrix that when multiplied by A yields A?

$$AI = A$$

- Yes! It is called the 'Identity Matrix' $I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

$$I\,\vec{x} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \vec{x}$$

Systems of equations

- Square matrices are very useful
- How do we solve this simple equation? divide both sides by a

$$ax = c \qquad\qquad x = a^{-1}c$$

- Now let's consider a 'system' of equations

$$\begin{aligned} x - 2y &= 3 \\ 3x + y &= 5 \end{aligned}$$

$$A\vec{x} = \vec{c}$$

where

- We can write this as:

$$\begin{pmatrix} 1 & -2 \\ 3 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 3 \\ 5 \end{pmatrix}$$

$$A = \begin{pmatrix} 1 & -2 \\ 3 & 1 \end{pmatrix}$$

$$\vec{c} = \begin{pmatrix} 3 \\ 5 \end{pmatrix}$$

Systems of equations

- How do we divide both sides by A? $A\vec{x} = \vec{c}$
- We can't, but we can multiply both sides by something which makes the A go away!
- The matrix inverse of A, denoted A^{-1}, has the property that:

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

- Thus to solve the system of equations $A\vec{x} = \vec{c}$

  - Multiply both sides of the eqn by A^{-1}

$$A^{-1}A\,\vec{x} = A^{-1}\vec{c}$$

$$I\,\vec{x} = A^{-1}\vec{c}$$

$$\Rightarrow \quad \boxed{\vec{x} = A^{-1}\vec{c}}$$

Matrix inverse in 2d

- For a matrix A given by $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$

the inverse of A is given by

$$A^{-1} = \frac{1}{\det(A)} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

where the determinant is given by:

$$\det(A) = ad - bc$$

$$A^{-1}A = \frac{1}{\det(A)} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

$$= \frac{1}{\det(A)} \begin{pmatrix} ad - bc & db - bd \\ -ca + ac & -cb + ad \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

The matrix has an inverse iff $\det(A) \neq 0$

The matrix is 'singular' if $\det(A) = 0$

Matrix inverse in 2d

- Back to our system of equations

$$A\vec{x} = \vec{c} \qquad A = \begin{pmatrix} 1 & -2 \\ 3 & 1 \end{pmatrix} \qquad \vec{c} = \begin{pmatrix} 3 \\ 5 \end{pmatrix}$$

- The determinant is $\det(A) = 1 - (-6) = 7$ so there is an inverse.

$$A^{-1} = \frac{1}{7}\begin{pmatrix} 1 & 2 \\ -3 & 1 \end{pmatrix}$$

- Thus,

$$\vec{x} = A^{-1}\vec{c} = \frac{1}{7}\begin{pmatrix} 1 & 2 \\ -3 & 1 \end{pmatrix}\begin{pmatrix} 3 \\ 5 \end{pmatrix} = \frac{1}{7}\begin{pmatrix} 13 \\ -4 \end{pmatrix}$$

Learning Objectives for Lecture 15

- Perceptrons and perceptron learning rule
- Neuronal logic, linear separability, and invariance
- Two-layer feedforward networks
- Matrix algebra review
- Matrix transformations

Matrix transformation

- You can see from our system of equations that the matrix A 'transformed' vector x into the vector c

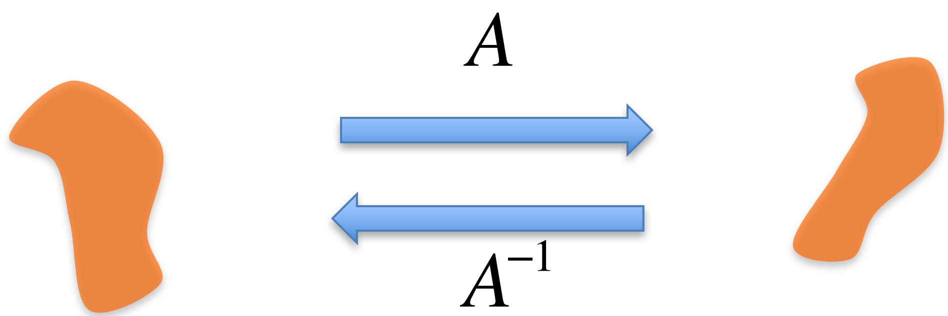
$$\vec{x} = \begin{pmatrix} 13/7 \\ -4/7 \end{pmatrix} \qquad \xrightarrow{\vec{c} = A\vec{x}} \qquad \vec{c} = \begin{pmatrix} 3 \\ 5 \end{pmatrix}$$

- The matrix A^{-1} transformed vector c back into the vector x

$$\vec{x} = \begin{pmatrix} 13/7 \\ -4/7 \end{pmatrix} \qquad \xleftarrow{\vec{x} = A^{-1}\vec{c}} \qquad \vec{c} = \begin{pmatrix} 3 \\ 5 \end{pmatrix}$$

Matrix transformation

- In general A maps the set of vectors in $\mathbb{R}^2$ onto another set of vectors in $\mathbb{R}^2$. What do these mappings look like?



$$\det(A) \neq 0$$

Matrix transformation

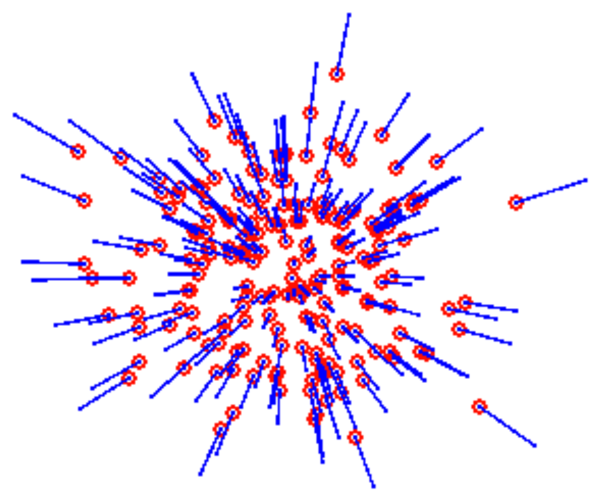
- We already know the simplest transformation, when A=identity

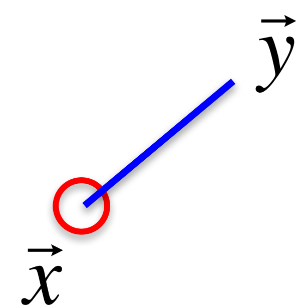
$$\vec{y} = I\vec{x} = \vec{x}$$

It is instructive to consider small perturbations from the identity matrix.

$$\vec{y} = A\vec{x} \qquad A = I + \Delta = \begin{pmatrix} 1+\delta & 0 \\ 0 & 1+\delta \end{pmatrix}$$

```
x=randn(2,N1); % Gaussian
delta=0.3;
I=[1,0;0,1];
A=I+[delta,0;0,delta];
y=A*x;
```





Matrix transformation

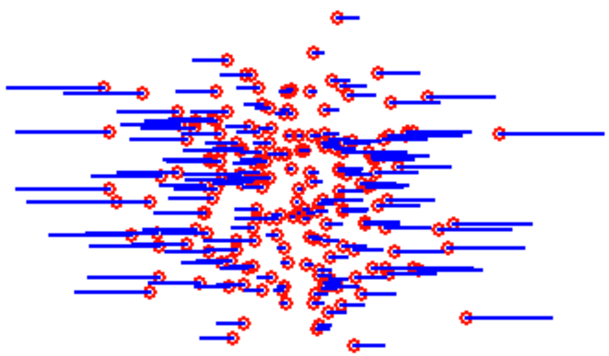
- It is instructive to consider small perturbations from the identity matrix.

For example…

$$\vec{y} = A\vec{x}$$

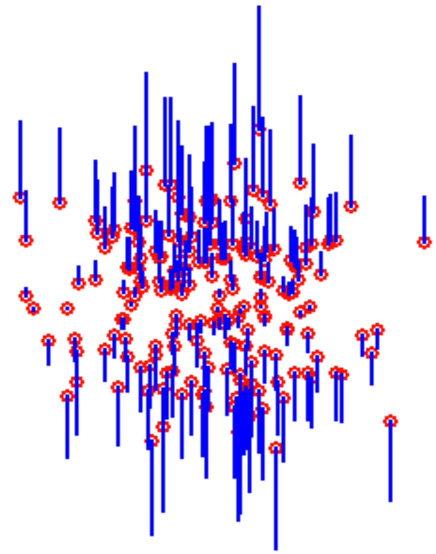
Stretch in x-direction

$$A = \begin{pmatrix} 1+\delta & 0 \\ 0 & 1 \end{pmatrix}$$



Stretch in y-direction

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 1+\delta \end{pmatrix}$$



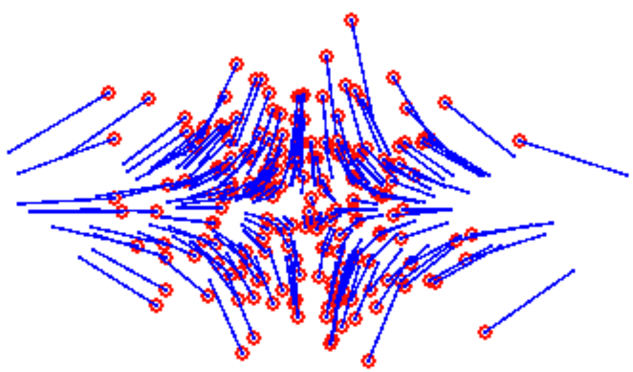
Matrix transformation

- It is instructive to consider small perturbations from the identity matrix.

For example…

$$\vec{y} = A\vec{x}$$

$$A = \begin{pmatrix} 1+\delta & 0 \\ 0 & 1-\delta \end{pmatrix}$$



Matrix symmetries

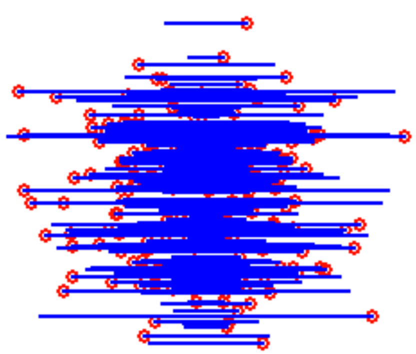
- Matrix multiplication can be used to produce 'symmetry operations'

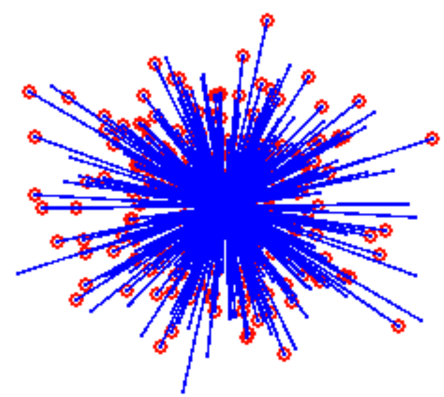
Mirror reflection

$$A = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$

Inversion through the origin

$$A = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$$

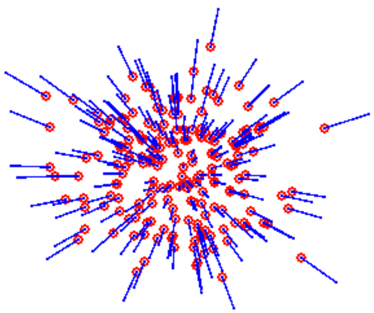




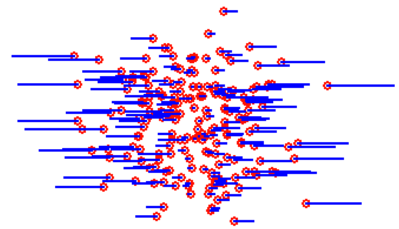
Matrix transformations $\vec{y} = A\vec{x}$

- Perturbations from the identity matrix

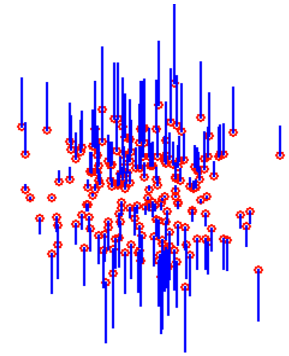
$$A = \begin{pmatrix} 1+\delta & 0 \\ 0 & 1+\delta \end{pmatrix}$$



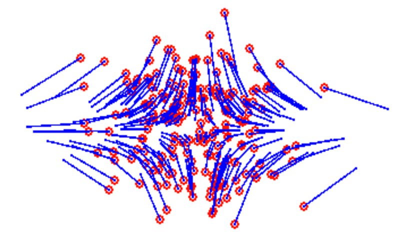
$$A = \begin{pmatrix} 1+\delta & 0 \\ 0 & 1 \end{pmatrix}$$



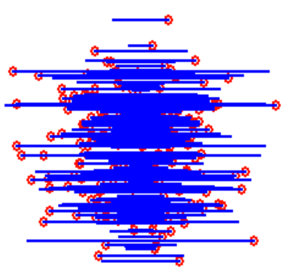
$$A = \begin{pmatrix} 1 & 0 \\ 0 & 1+\delta \end{pmatrix}$$



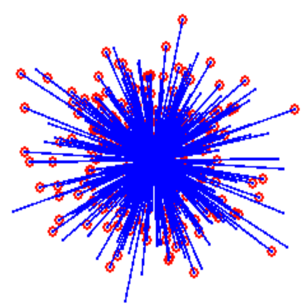
$$A = \begin{pmatrix} 1+\delta & 0 \\ 0 & 1-\delta \end{pmatrix}$$



$$A = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$



$$A = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$$



These are all diagonal matrices

$$\Lambda = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix}$$

$$\Lambda^{-1} = \begin{pmatrix} a^{-1} & 0 \\ 0 & b^{-1} \end{pmatrix}$$

Matrix transformation

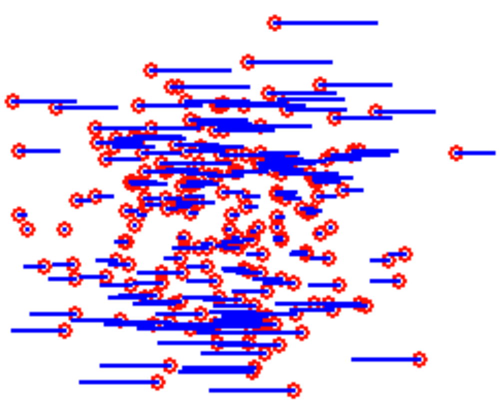
- It is instructive to consider small perturbations from the identity matrix.

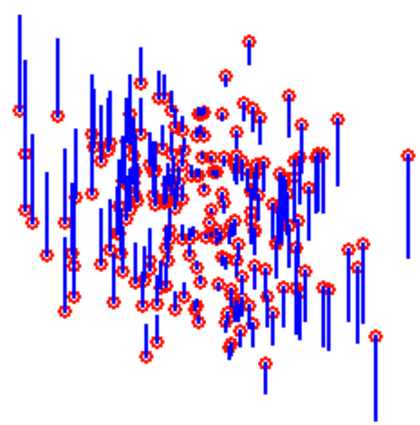
Shear along x

$$A = \begin{pmatrix} 1 & +\delta \\ 0 & 1 \end{pmatrix}$$

Shear along y

$$A = \begin{pmatrix} 1 & 0 \\ -\delta & 1 \end{pmatrix}$$



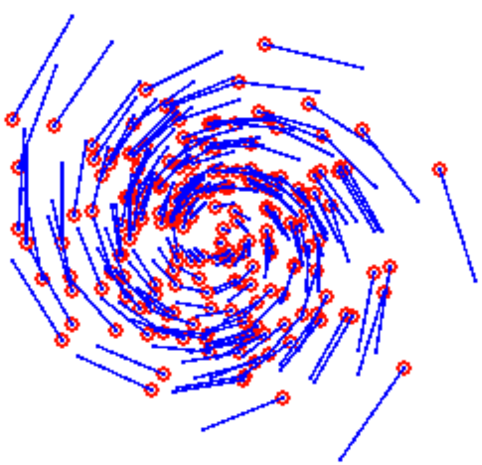


Matrix transformation

- It is instructive to consider small perturbations from the identity matrix.

Rotation!

$$A = \begin{pmatrix} 1 & +\delta \\ -\delta & 1 \end{pmatrix}$$

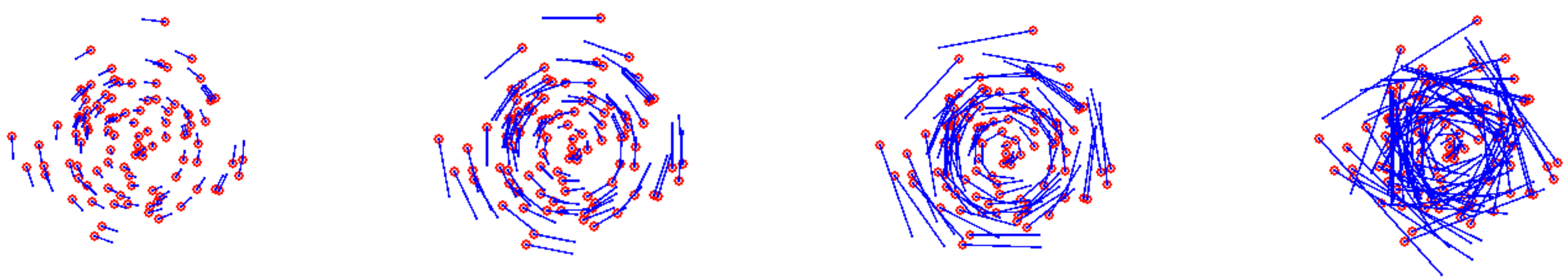


Rotation matrix

- We can implement a rotation in the plane by an arbitrary angle θ with the following matrix.

$$\Phi(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \qquad \Phi(45^o) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$$

$\theta = 10^o$ $\qquad \theta = 25^o$ $\qquad \theta = 45^o$ $\qquad \theta = 90^o$



Rotation matrix

- Does a rotation matrix have an inverse? $\det(\Phi) = 1$

$$\Phi(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \qquad \Phi(-\theta) = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

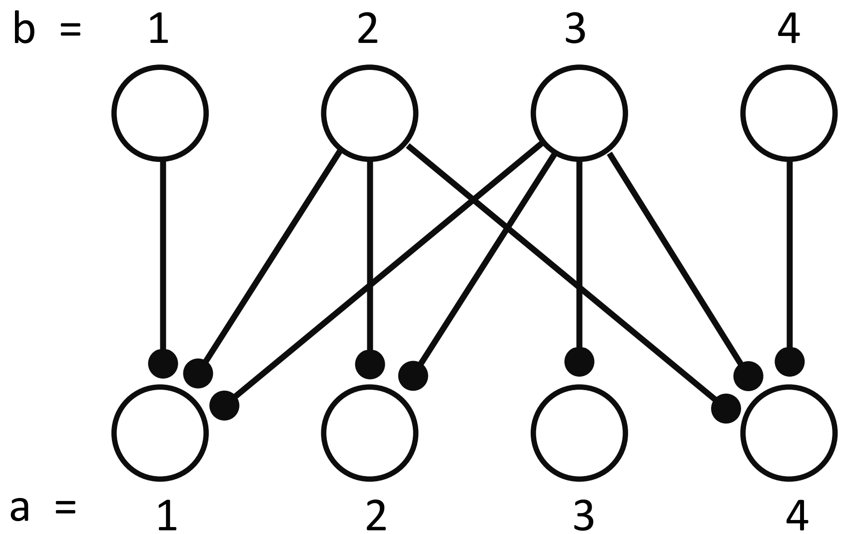
- A rotation by angle +θ followed by a rotation by angle –θ just puts everything back where it was.

$$\Phi(-\theta)\Phi(\theta) = I \qquad \Phi^{-1}(\theta) = \Phi(-\theta)$$

- Also, the inverse of A is just the transpose of A!

$$\Phi^{-1}(\theta) = \Phi^{T}(\theta)$$

Two-layer feed-forward network



- Our feed-forward network implements an arbitrary matrix transformation!

$$\vec{v} = W \vec{u}$$

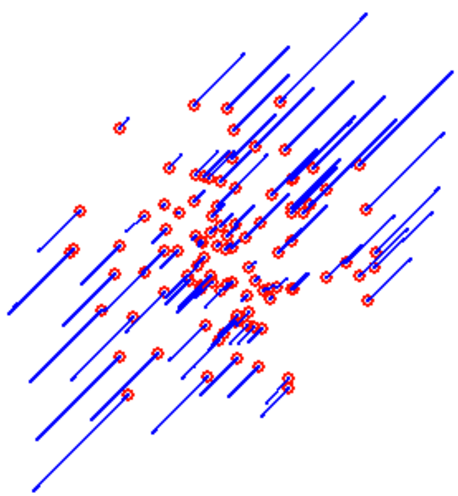
Learning Objectives for Lecture 15

- Perceptrons and perceptron learning rule
- Neuronal logic, linear separability, and invariance
- Two-layer feedforward networks
- Matrix algebra review
- Matrix transformations

Rotated transformations

- The rotation matrix allows us to do a very cool trick.
- We can do any of the transformations above (stretch, mirror reflection, shear), not just along the axes, but in any arbitrary direction.

For example, stretch along a 45° angle



Rotated transformations

- We will do this by making three successive transformations:

  'Unrotate' our vectors by angle $-\theta$: $\Phi(-\theta) = \Phi^T(\theta)$

  Make a transformation : Λ

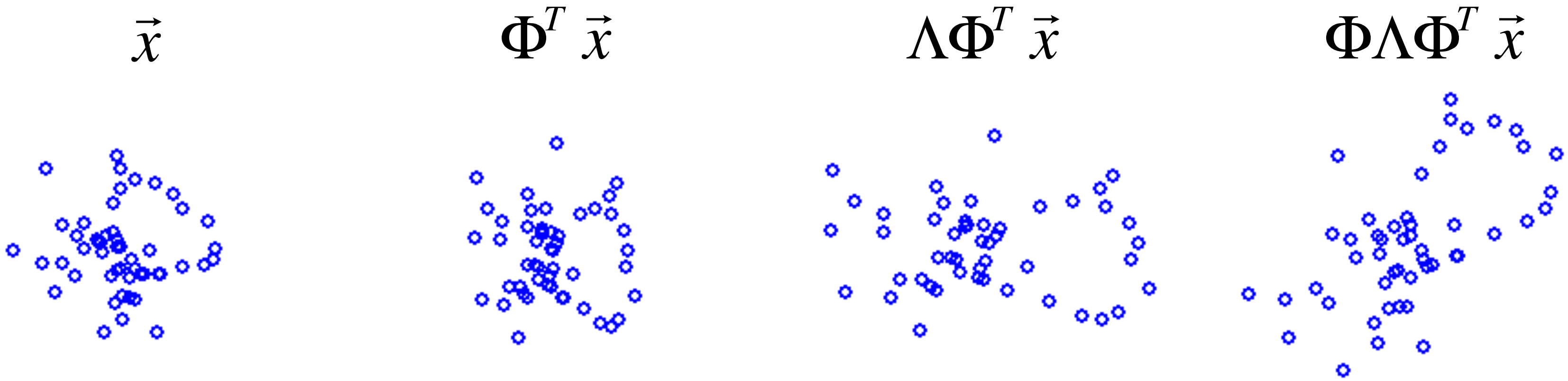
  Then rotate our vectors back by angle θ: $\Phi(\theta)$

- We do each of these steps by multiplying our matrices together

$$\Phi \; \Lambda \, \Phi^T \, \vec{x}$$

Rotated transformations

- Let's construct a matrix that produces a stretch along a 45° angle…



$$\Phi^T = \Phi(-45^o) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$$

$$\Lambda = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\Phi(+45^o) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$$

$$\Phi\Lambda\Phi^T = \frac{1}{2}\begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}$$