Introduction to Neural Computation

Prof. Michale Fee
MIT BCS 9.40 — 2018

Lecture 14
Rate models and Perceptrons

Game plan for Lectures 14 – 18

Examine the computational properties of networks of neurons

- Rate models
- Feed-forward neural networks (Perceptrons)

- Matrix operations
- Basis sets
- Principal components analysis

- Recurrent Neural Networks
- Line attractors in short term memory
- Hopfield networks

Learning Objectives for Lecture 14

- Derive a mathematically tractable model of neural networks
  (the rate model)

- Building receptive fields with neural networks

- Vector notation and vector algebra

- Neural networks for classification

- Perceptrons

Neural Network Models

- We are going to examine some of the computational properties of networks of neurons.

- Our first step is to derive a simplified mathematical model of neurons that we can study analytically.

Why would we want to do this?

- For example, we approximated the detailed spiking properties of neurons (HH model) with an integrate & fire (IF) model.

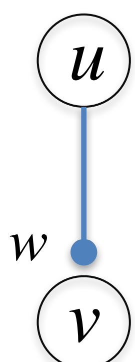
- This is not enough of a simplification to develop an analytical model of neural circuits.

Rate models

- Let’s start with two neurons, an input neuron that synapses with weight w onto an output neuron.
- We are going to ignore spike times, and describe the inputs and outputs of our neurons simply as firing rates.
- In the simplest case… linear neurons! $v = wu$

How can we justify this?

input neuron



output neuron

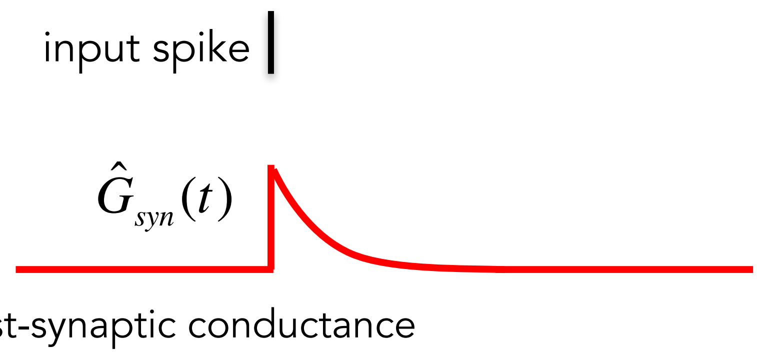
$u =$ firing rate of input neuron

$w =$ synaptic strength (weight)

$v =$ firing rate of output neuron

Rate models

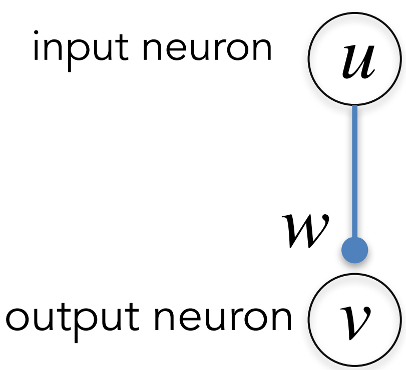
- Let's examine the response of the output neuron to a single input spike.



$$\hat{G}_{syn}(t) = G_{\max}\, e^{-t/\tau_s}$$

$$\hat{I}_{syn}(t) = \hat{G}_{syn}(t)[\cancel{V - E_{syn}}]$$

Ignore synaptic saturation



Now write $\hat{I}_{syn}(t) = wK(t)$

where $K(t)$ is a kernel of unit area

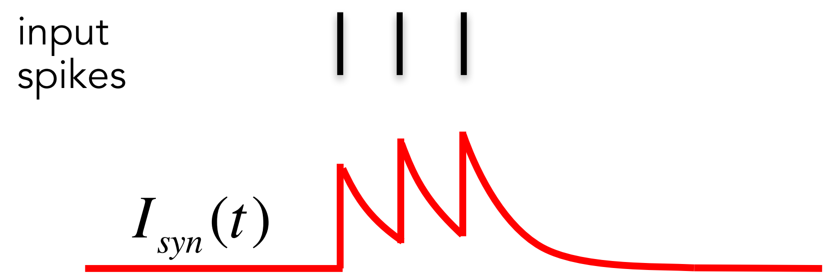
$K(t) = \dfrac{1}{\tau_s} e^{-t/\tau_s}$ area $=1$

Rate models

- How do we get the response to multiple input spikes?

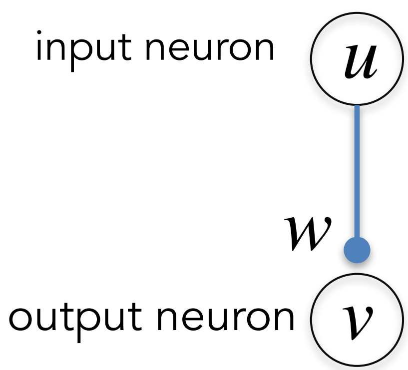
Input spike train

$$\rho(t) = \sum_i \delta(t - t_i)$$



- We convolve the input spike train with the synaptic kernel !

$$I_{syn}(t) = w\,\underbrace{K * \rho(t)}_{\text{But what is this?}}$$



- If K is a kernel with area normalized to one, then…

$$K * \rho(t) \quad = u(t)$$

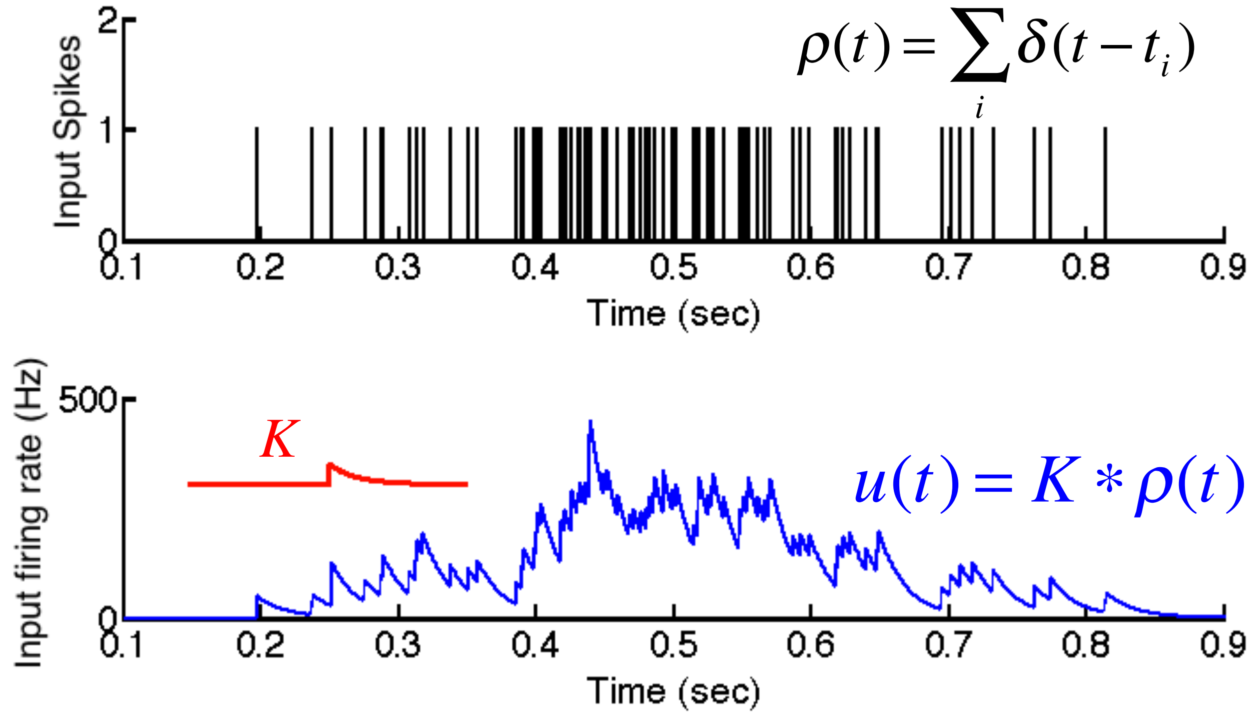
is just the firing rate of the input neuron!

Rate models

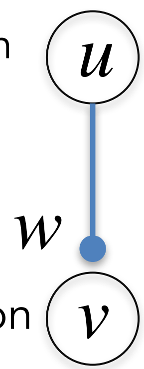
- Thus, we can write presynaptic firing rate as

pulse_of_firing_spikes.m

$$u(t) = K * \rho(t)$$



input neuron



output neuron

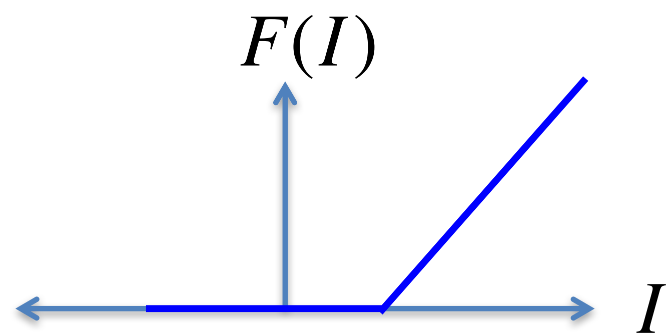
$$I_{syn}(t) = w K * \rho(t)$$
$$= w u(t)$$

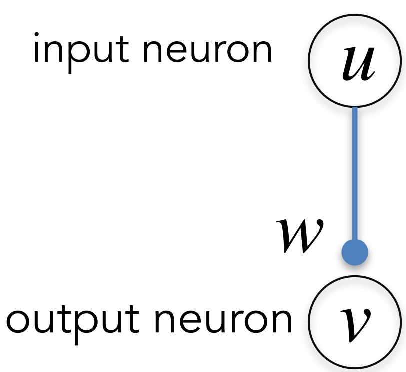
Rate models

- Now, how about the firing rate of the output neuron?

- You remember that when you injected a constant current into our Integrate and Fire model…

  The steady-state firing rate of the neuron had a threshold current below which the neuron would not spike, and some increasing f.r. above threshold.





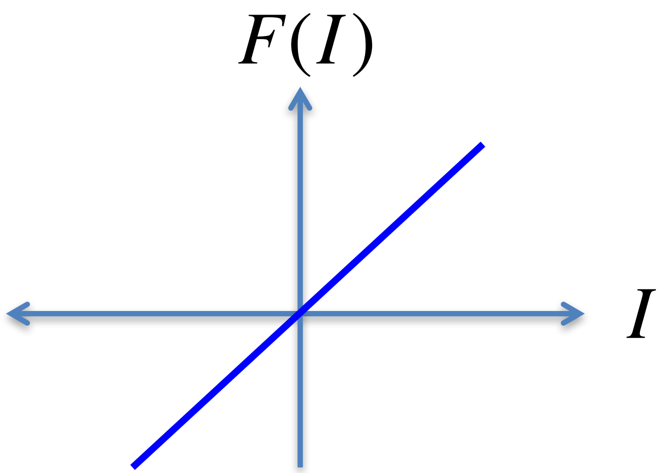
- In general, we can write the output firing rate of our model neuron as:

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

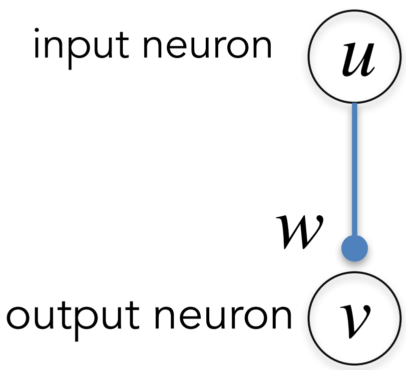
Linear rate models

- We will now consider an even greater simplification of our neurons… assume they are linear.

$$F[x] = x$$



We will come back to non-linear neurons shortly… they will be very important.



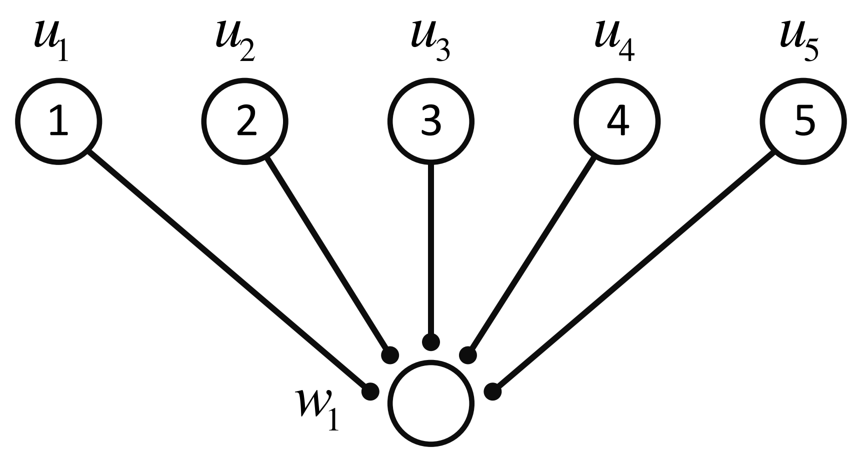
- Of course real neurons can't have negative firing rates, but we can gain a lot of insight using this approximation.

- Thus, we can write the output firing rate of our linear neuron as:

$$v = wu$$

Multiple inputs

- What happens when our output neuron has many inputs?



- The total input to our neuron is a sum of all the different inputs weighted by their synaptic strength!

$$I_{syn} = w_1 u_1 + w_2 u_2 + w_3 u_3 + \ldots$$

- The steady-state response of our linear neuron is now:

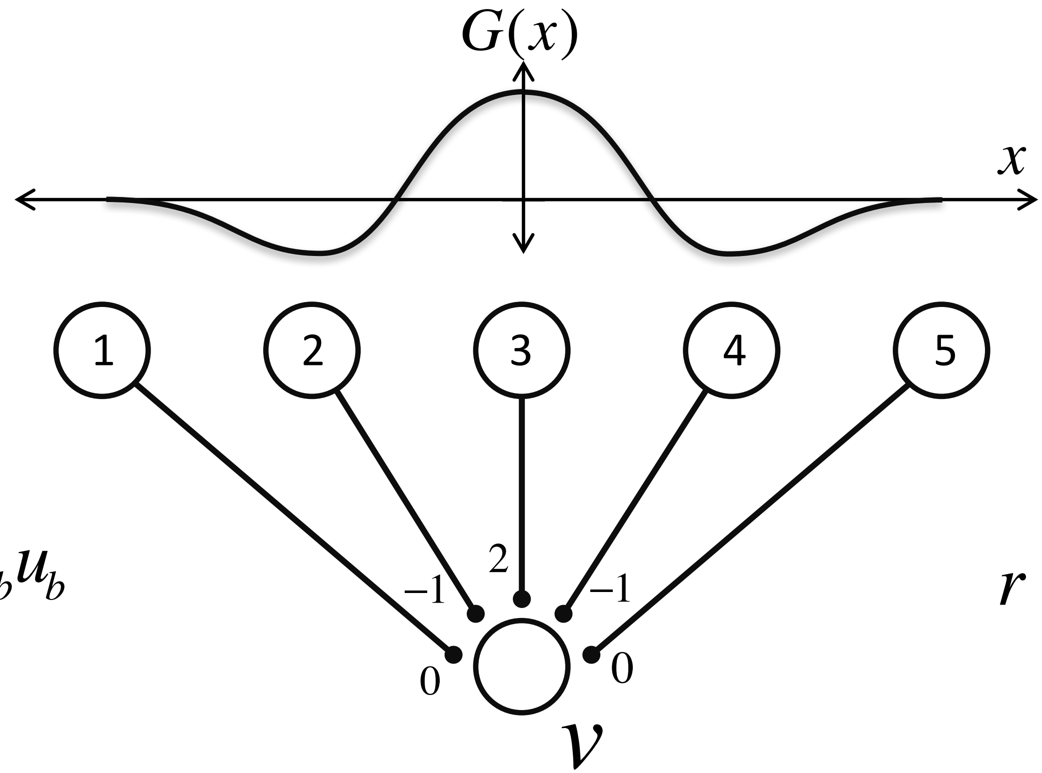
$$v = \sum_b w_b u_b$$

Learning Objectives for Lecture 14

- Derive a mathematically tractable model of neural networks (the rate model)
- Building receptive fields with neural networks
- Vector notation and vector algebra
- Neural networks for classification
- Perceptrons

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



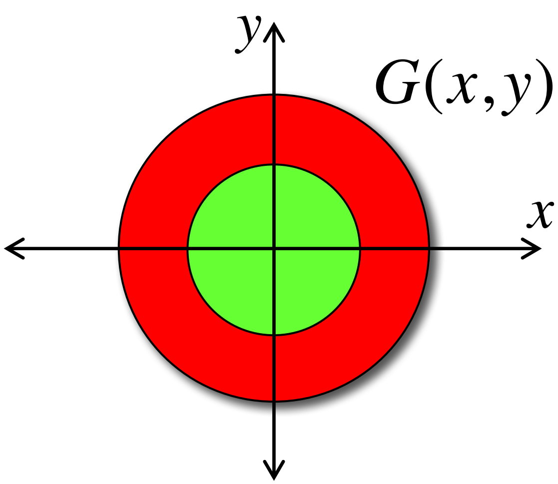
$$v = \sum_b w_b u_b$$

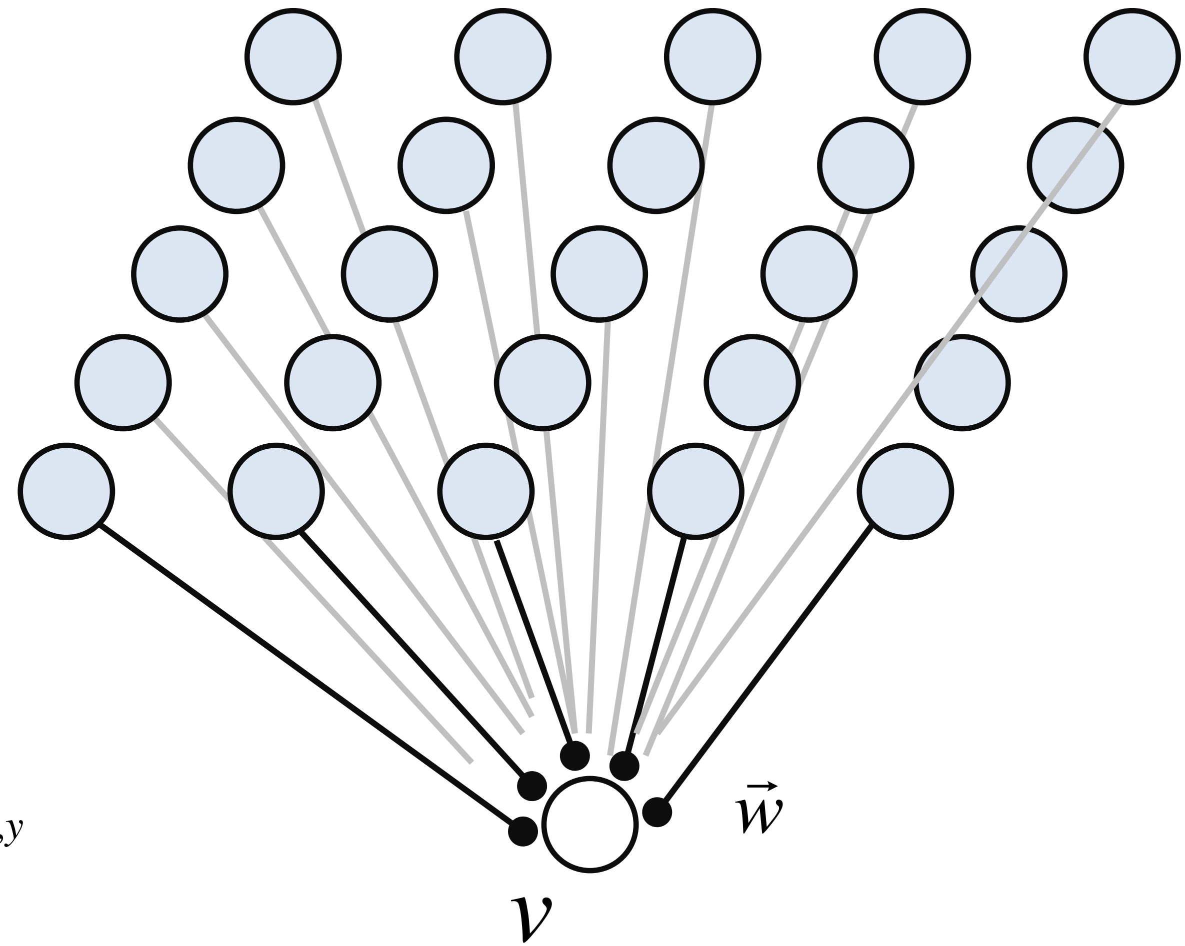
$$r = \int G(x) I(x)$$

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

How to build a receptive field

- We can even build 2D receptive fields with the same formalism.



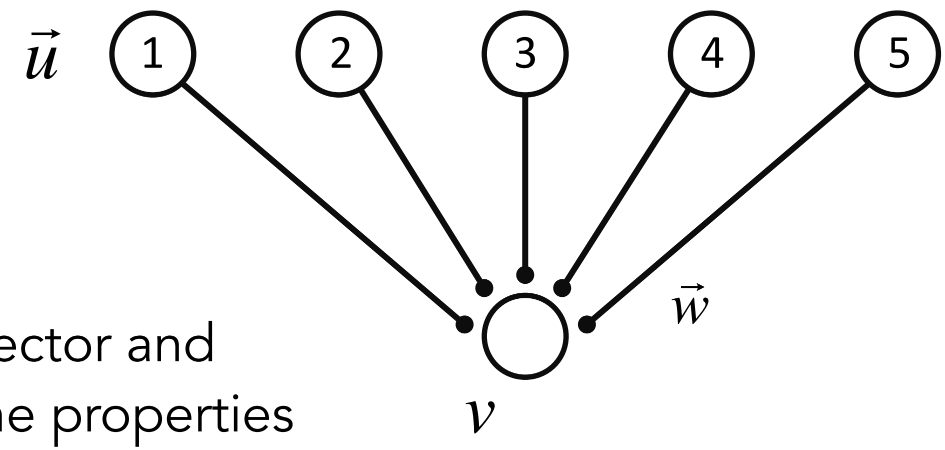


$$v = \sum_{x,y} w_{x,y} u_{x,y}$$

Linear algebra detour

- Mathematically, we have described the response of our linear neuron as

$$v = \sum_b w_b u_b$$



- We are going to start using vector and matrix notation to describe the properties of networks

- …because it is much more compact and powerful

- We have to take a short detour to learn some linear algebra.

Learning Objectives for Lecture 14

- Derive a mathematically tractable model of neural networks
  (the rate model)
- Building receptive fields with neural networks
- Vector notation and vector algebra
- Neural networks for classification
- Perceptrons

Linear algebra detour

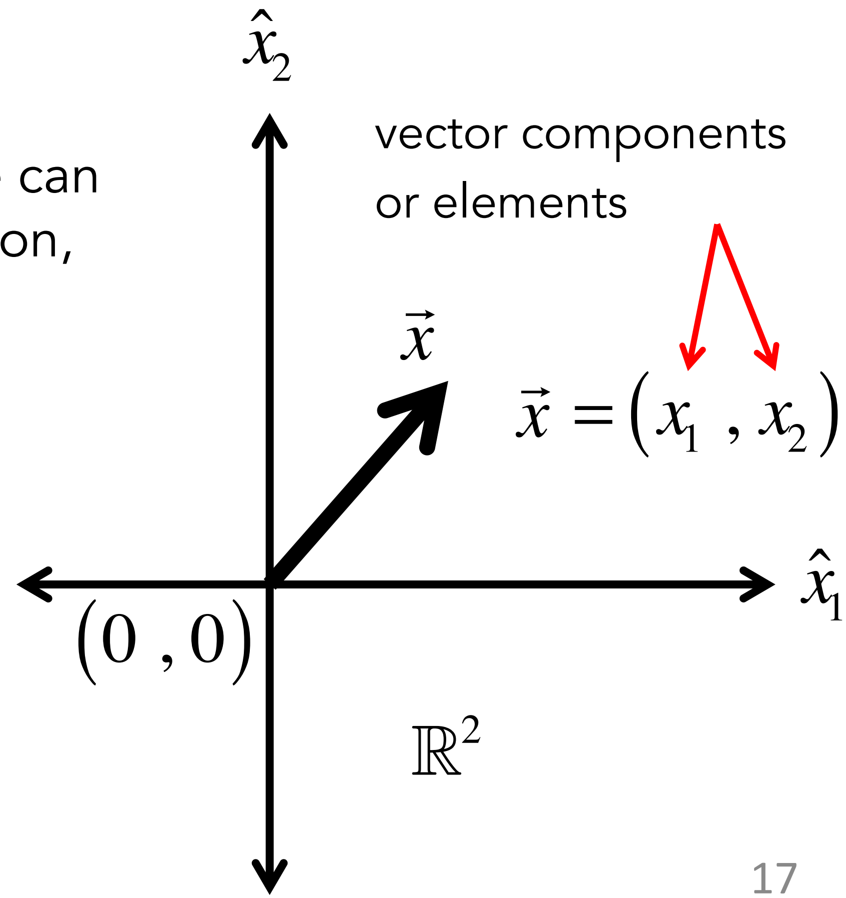
- A vector is a collection of numbers.

- The number of numbers in the collection is called the dimensionality of the vector.

- If there are two or three numbers, we can draw a vector as a position, or direction, in space.

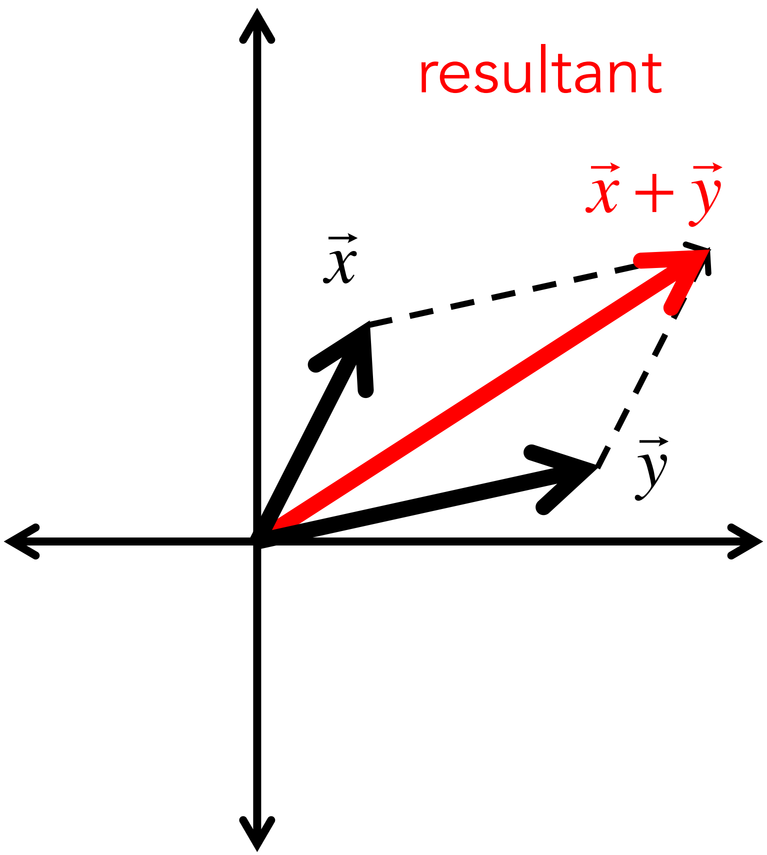
$$\vec{x} = \begin{pmatrix} x_1 & x_2 \end{pmatrix}$$ row vector

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$ column vector



Vector sum

- Sum of two vectors



$$\vec{x}+\vec{y} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} + \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix} = \begin{pmatrix} x_1+y_1 \\ x_2+y_2 \\ \dots \\ x_n+y_n \end{pmatrix}$$

Element-by-element addition

Vector products

- There are several ways of taking the product of two vectors
    - Element-by-element product
    - Inner product
    - Outer product

      We will cover this later
    - Cross product

      Important in physics, but we won't cover this.

Vector products

- Element-by-element product (Hadamard product)

$$\vec{x} \circ \vec{y} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \circ \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} x_1 y_1 \\ x_2 y_2 \\ \vdots \\ x_n y_n \end{pmatrix}$$

- In MATLAB®, the element-by-element product is x.*y

Vector products

- Inner product or dot product

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \qquad \vec{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

$$\vec{x} \cdot \vec{y} = x_1 y_1 + x_2 y_2 + \ldots + x_n y_n$$

$$= \sum_{i=1}^{n} x_i y_i = scalar$$

Some properties…

commutative

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

distributive

$$\vec{w} \cdot (\vec{x} + \vec{y}) = \vec{w} \cdot \vec{x} + \vec{w} \cdot \vec{y}$$

linearity

$$(a\vec{x}) \cdot \vec{y} = a(\vec{x} \cdot \vec{y})$$

Vector products

- Inner product in matrix notation

$$\vec{x}\cdot\vec{y} = \underset{1 \times N}{\begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix}} \underset{N \times 1}{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}} = \underset{1 \times 1}{scalar}$$

- In MATLAB®…

```
x = [1; 2; 3];        % column vector (1 x 3)
y = [2; 4; 6];        % column vector (1 x 3)
z = x' * y;           % ' means transpose
                      % row*column vector (1x3)*(3x1)
```

Vector products

- Dot product of a vector with itself

$$\vec{x} \cdot \vec{x} = \sum_{i=1}^{n} x_i \, x_i \;\; = \left|\vec{x}\right|^2$$

$\left|\vec{x}\right|$ is the 'norm' or 'magnitude' of the vector

$$\left|\vec{x}\right| = \sqrt{\sum_{i=1}^{n} x_i \, x_i}$$

Pythagorean theorem

Unit vector

- A unit vector has length 1.

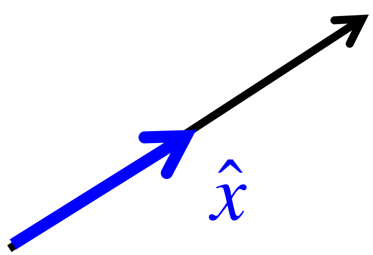
$$|\hat{x}| = 1 \qquad\qquad \hat{x} \cdot \hat{x} = 1$$

- We can make a unit vector out of any vector

$$\hat{x} = \frac{1}{|\vec{x}|}\vec{x}$$

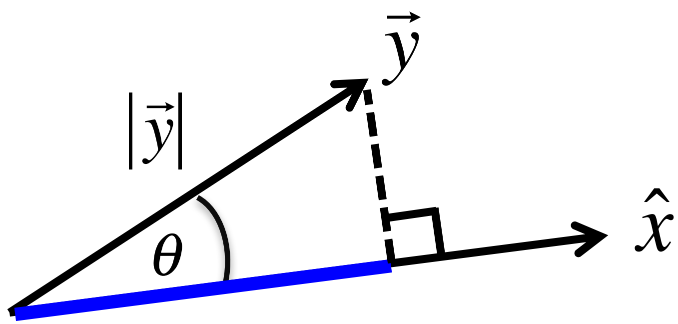
- We can express any vector as a product of a length times a unit vector:

$$\vec{x} = |\vec{x}|\,\hat{x}$$



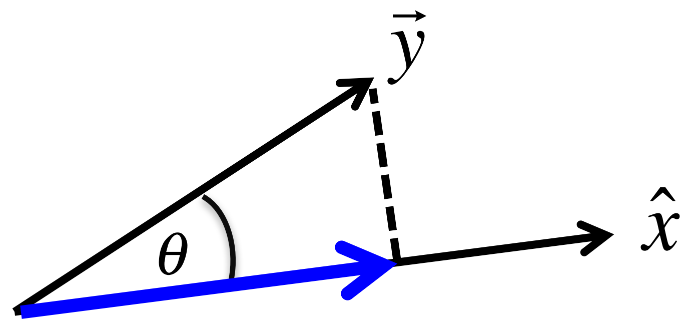
Projection

Find the component of vector $\vec{y}$ in the direction of vector $\hat{x}$. Let $\hat{x}$ be a unit vector.



$\vec{y} \cdot \hat{x}$

$= |\vec{y}| \cos\theta$



$(\vec{y} \cdot \hat{x})\hat{x}$

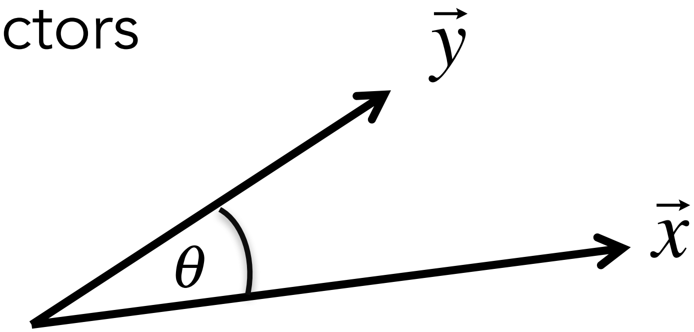
= scalar times unit vector

'Scalar projection' of $\vec{y}$ onto $\hat{x}$

'Vector projection' of $\vec{y}$ onto $\hat{x}$

Geometric intuition of dot products

- Dot product is related to the cosine of the angle between two vectors



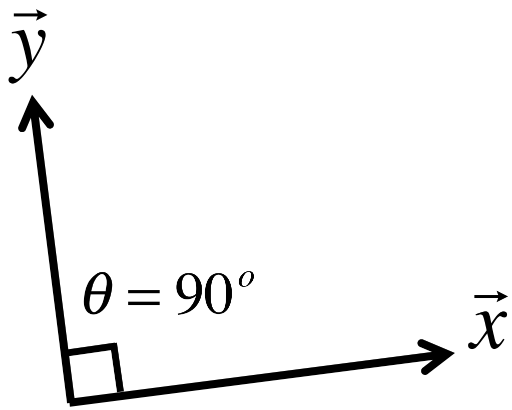
$$\vec{x} \cdot \vec{y} = |\vec{x}||\vec{y}|\cos\theta \qquad \cos\theta = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}||\vec{y}|}$$

- If x and y are unit vectors, then…

$$\hat{x} \cdot \hat{y} = \cos\theta$$

Orthogonality

- Two vectors are orthogonal (perpendicular) if and only if their dot product is zero.



$$\vec{x} \cdot \vec{y} = |\vec{x}||\vec{y}|\cos\theta$$

$$\vec{x} \cdot \vec{y} = 0$$

$$\cos 90^o = 0$$

- The projection of y onto x is zero.

- The vector projection of y along x is the zero vector.

‘Correlation’ intuition of dot product

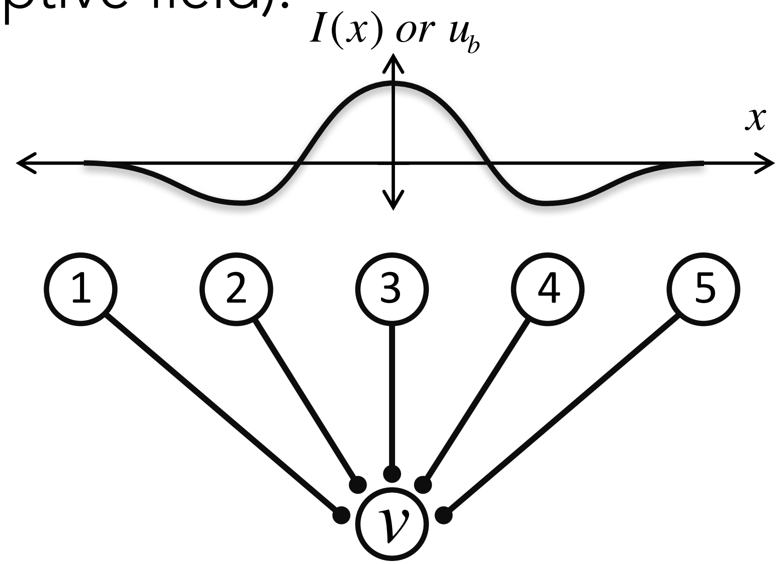
- The dot product is related to the statistical correlation between the elements of the two vectors

$$\cos\theta = \frac{\vec{y}\cdot\vec{x}}{|\vec{x}||\vec{y}|} = \frac{\sum_i y_i x_i}{\sqrt{\sum_i x_i x_i}\sqrt{\sum_i y_i y_i}}$$

Bounded between -1 and 1

Optimal stimulus

- The response of a neuron is the dot-product of the stimulus vector with the weight vector (receptive field).



$$v = \sum_b w_b u_b$$

$$v = \vec{w} \cdot \vec{u} = |\vec{w}||\vec{u}|\cos\theta$$

- Thus, for a given amount of power in the stimulus $a^2 = |\vec{u}|^2$, the stimulus that has the best overlap with the receptive field ($\cos\theta = 1$) produces the largest neuronal response.

- We now have a definition of the 'optimal stimulus':

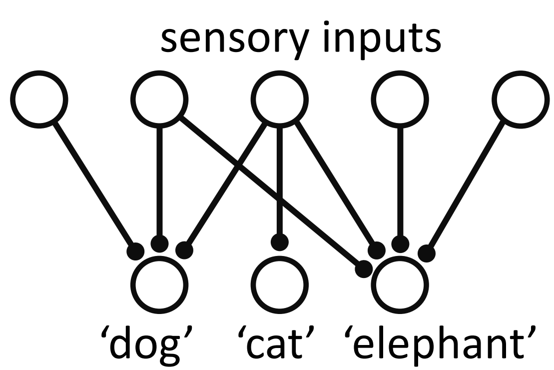
$$\vec{u} = a\hat{w}$$

Learning Objectives for Lecture 14

- Derive a mathematically tractable model of neural networks
  (the rate model)
- Building receptive fields with neural networks
- Vector notation and vector algebra
- Neural networks for classification
- Perceptrons

Classification

- A general computational problem solved by brain circuits is that of classification.
- Does that visual input represent a house cat or a tiger
    - an edible object or a poisonous one
    - a friendly dog or a wolf
- Feedforward circuits can be very good at classification



Object recognition in human cortex

Figure removed due to copyright restrictions. See Lecture 14 video or Figure 1 in Quiroga, R.Q., et al. "Invariant Visual Representation by Single Neurons in the Human Brain." *Nature* 435 (2005): 1102-1107.

Recordings in Right Anterior Hippocampus

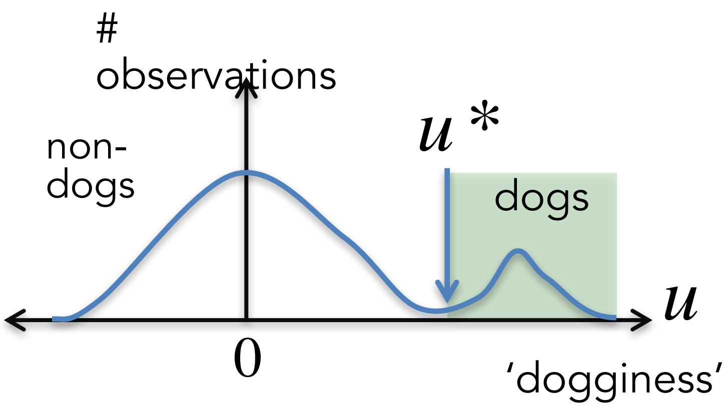
Quiroga et al Nature 2005

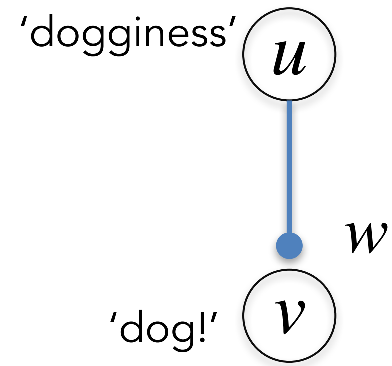
Learning Objectives for Lecture 14

- Derive a mathematically tractable model of neural networks
  (the rate model)
- Building receptive fields with neural networks
- Vector notation and vector algebra
- Neural networks for classification
- Perceptrons

Perceptrons

- How do we make a neuron that fires when it sees a dog, but does not fire when there is no dog?
- Classification problem in one dimension: one input neuron whose firing rate is proportional to a feature - 'dogginess'.





- A central feature of classification is decision making.
  - -There exists a 'classification boundary' in stimulus space that separates dogs from non-dogs.
- How does a neural circuit make a decision? Spike threshold!

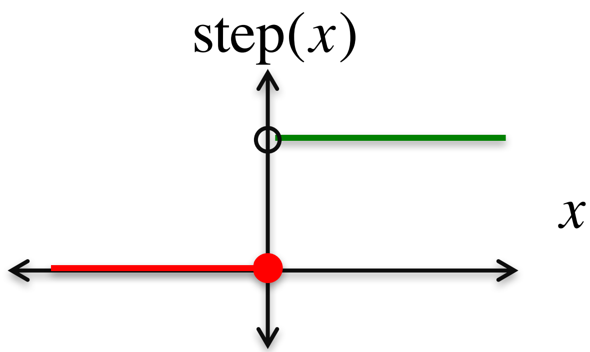
Generic images sources unknown. All rights reserved. This content is excluded from our Creative Commons license. For more information, see https://ocw.mit.edu/help/faq-fair-use/.

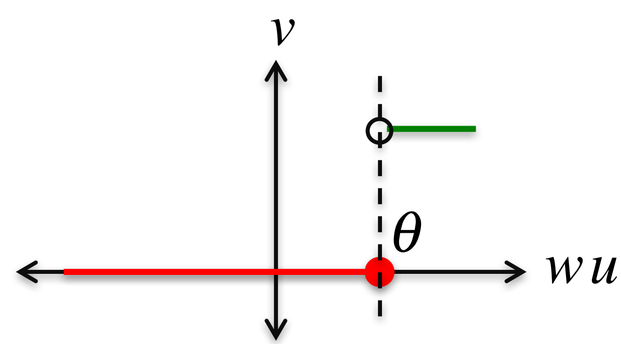
Binary threshold unit

- For a perceptron, we make a simplified model of a neuron that is very good at making decisions:

$$F(x) = \text{step}(x) \qquad\qquad v = F(wu - \theta)$$

Theta is the threshold, not an angle.





Neuron fires when the input $wu > \theta$

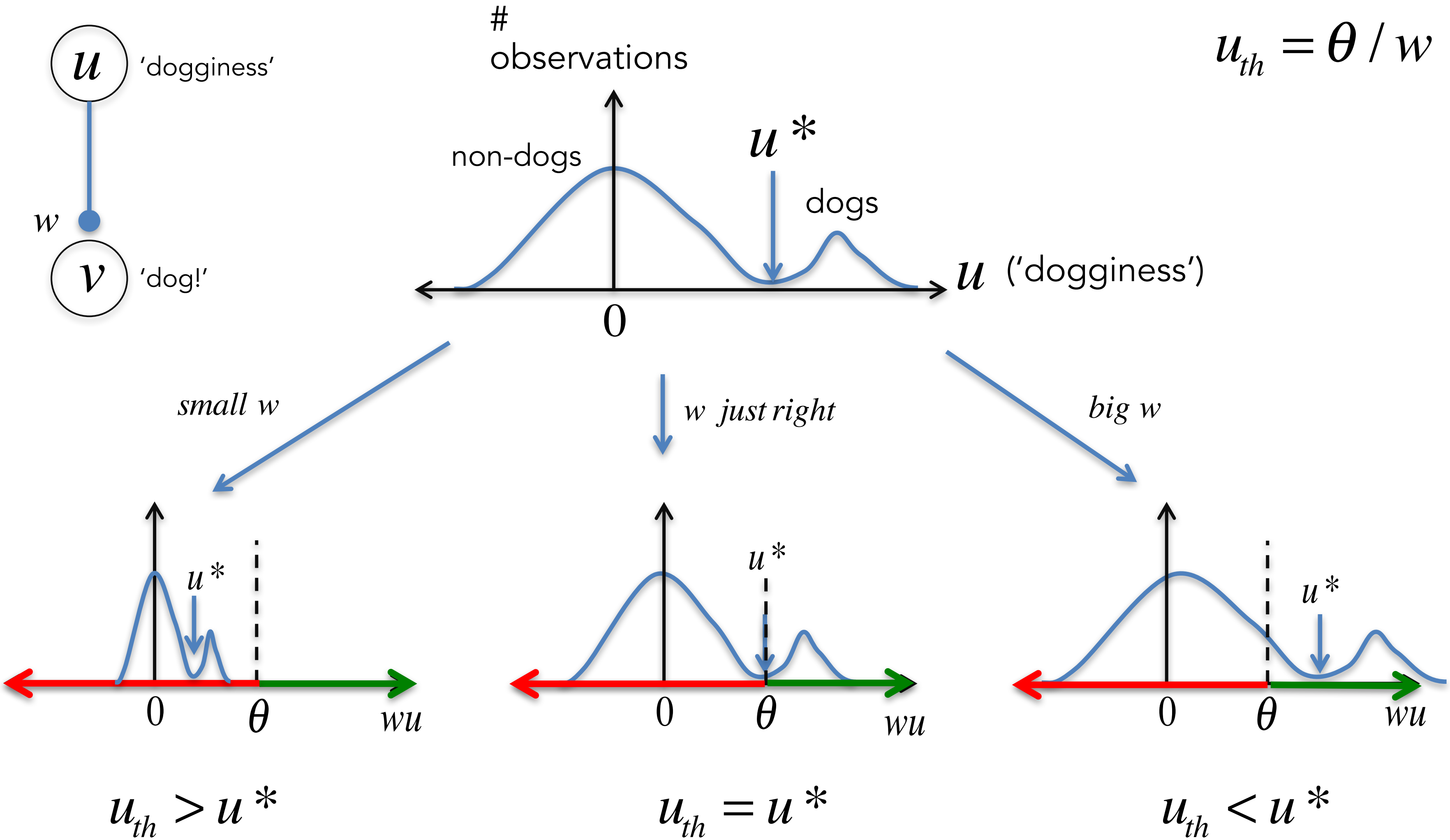
Thus, the output neuron begins to fire when the input neuron has a firing rate greater than the 'decision boundary.'

$$u_{th} = \theta / w$$

Setting the weight

- To classify, we need to learn the right w to make $u_{th} = u^*$

$$u_{th} = \theta / w$$



Setting the weight

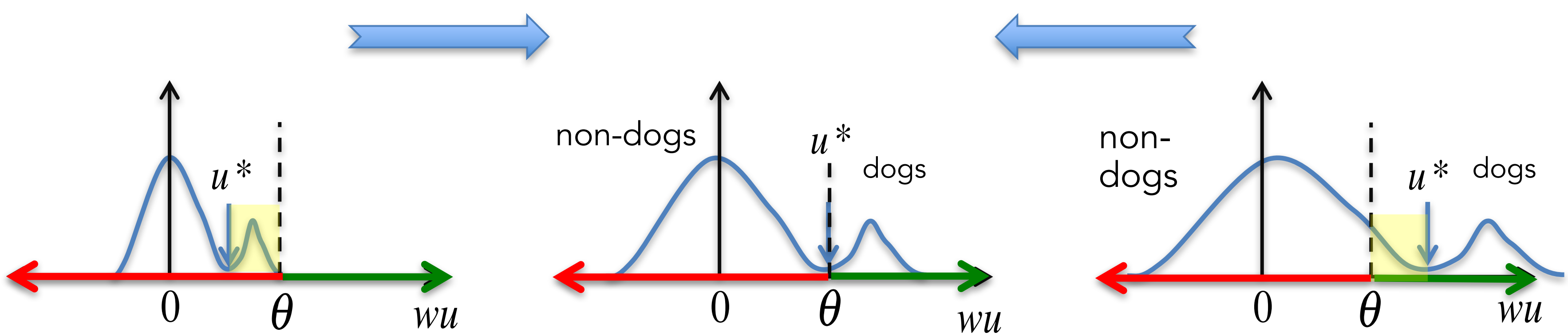
- To classify, we need to learn the right w to make $u_{th} = u^*$

Error: Dogs classified as non-dogs

$\Rightarrow$ Make w bigger

Error: Non-dogs classified as dogs

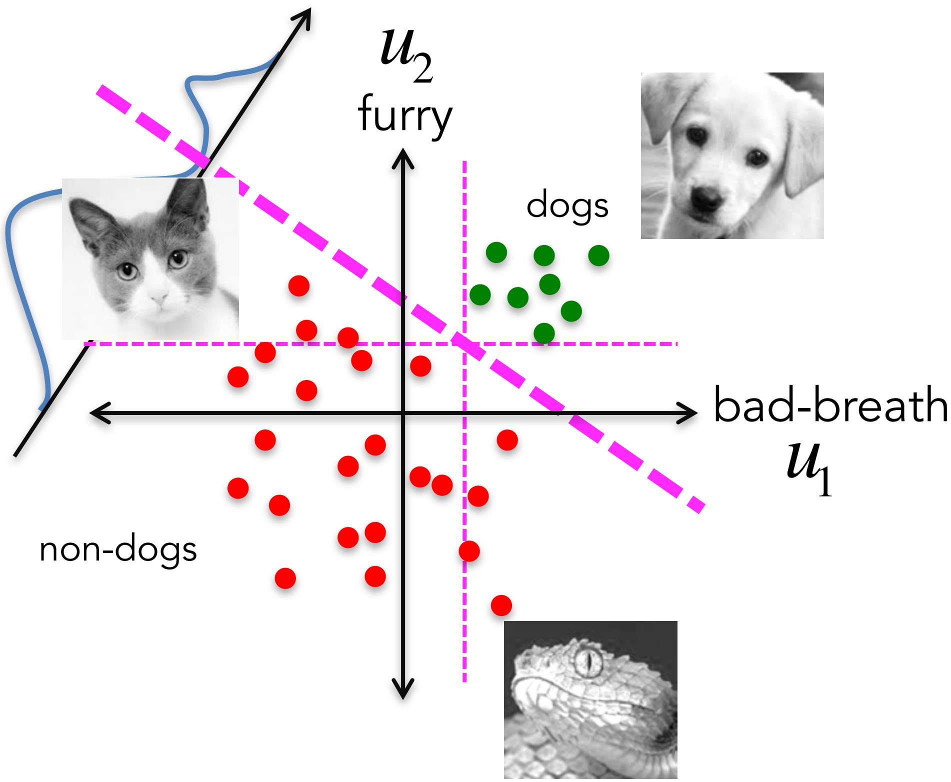
$\Rightarrow$ Make w smaller

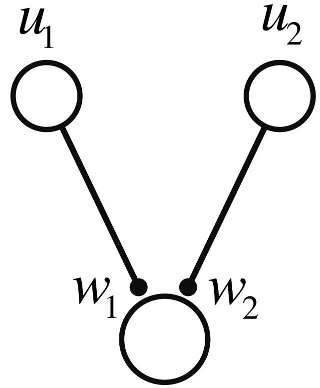


$u_{th} > u^*$ $\qquad$ $u_{th} = u^*$ $\qquad$ $u_{th} < u^*$

Decision boundary in two dimensions

- Sometimes classification has to be done on the basis of many features, not just one.





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

Generic images sources unknown. All rights reserved. This content is excluded from our Creative Commons license. For more information, see https://ocw.mit.edu/help/faq-fair-use/.

Decision boundary in two dimensions

- Let's look at the case where our neuron gets two inputs

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

- Now the decision boundary looks different…

$$\vec{w} \cdot \vec{u} - \theta = 0$$

what is this?

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

- This is an equation for a line in the space of $\vec{u}$, specified by the weights $\vec{w}$ and threshold θ.

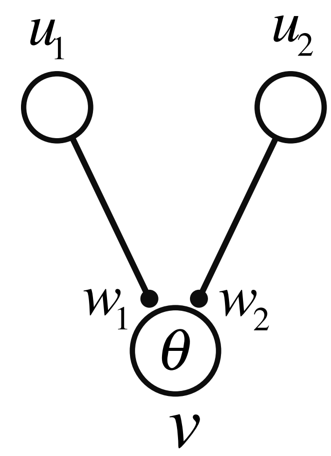
$$w_1 u_1 + w_2 u_2 = \theta$$

Decision boundary in two dimensions

- Let's start by looking at the case where $\theta = 0$

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

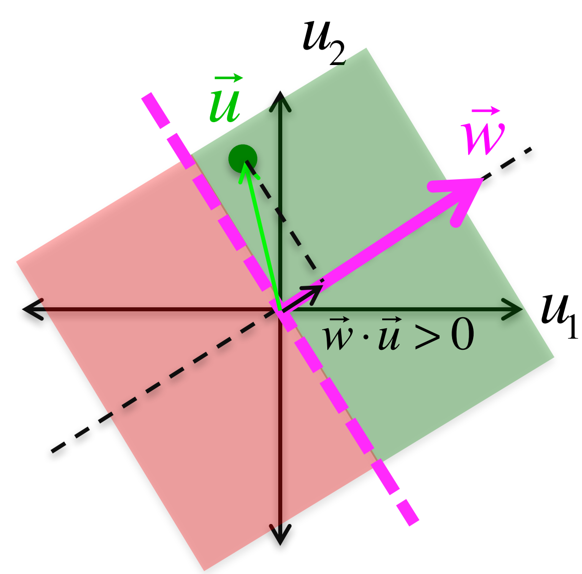
- The neuron now fires when the projection of $\vec{u}$ along $\vec{w}$ is positive $\vec{w} \cdot \vec{u} > 0$



- The decision boundary is given by

$$\vec{w} \cdot \vec{u} = 0$$

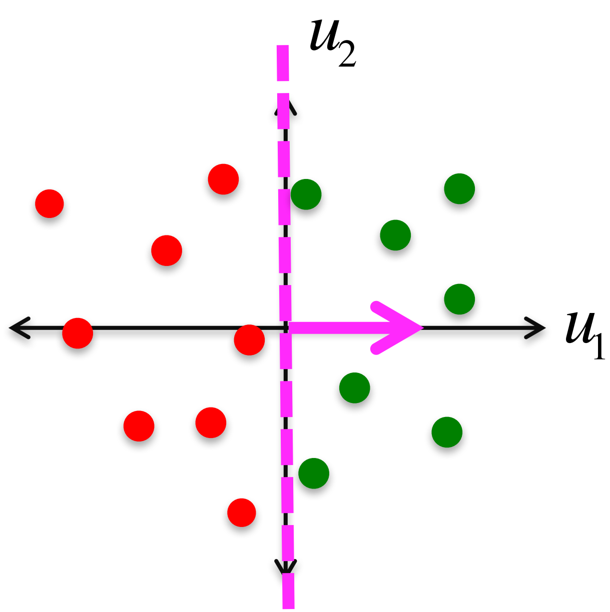
- This is the set of all vectors u that have zero projection along w.

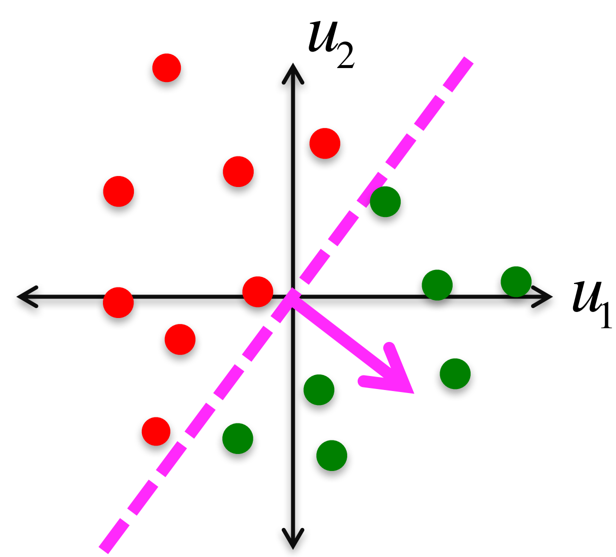


All vectors on a line going through the origin and perpendicular to w !

Classification in two dimensions

- Let's look at this for a few simple cases in two dimensions





$\vec{w} = (1\,,0)$

$\theta = 0$

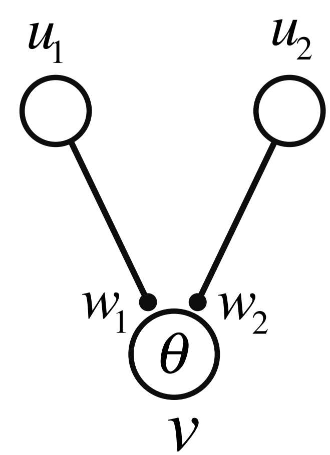
$\vec{w} = (1\,,-1)$

$\theta = 0$

Classification in two dimensions

- Now let's look at the case where $\theta \neq 0$

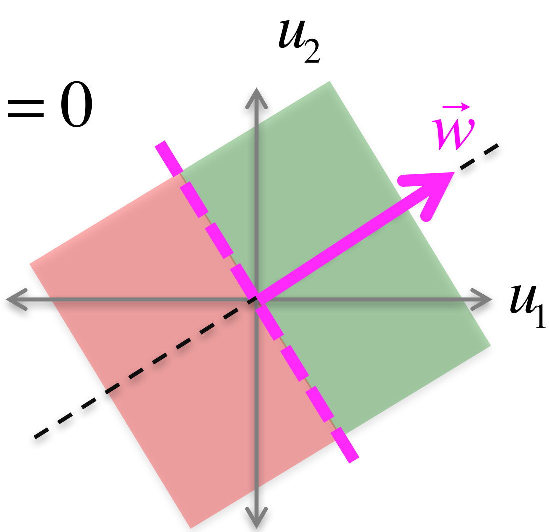
$$v = F(\vec{w} \cdot \vec{u} - \theta)$$



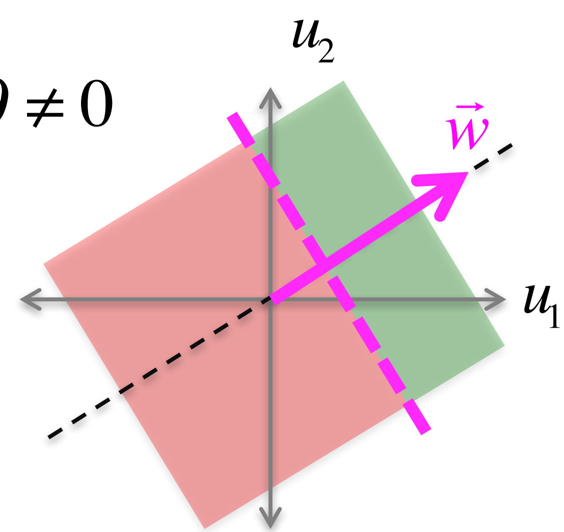
- Now the decision boundary is $\vec{w} \cdot \vec{u} = \theta$

- This is the set of all vectors $\vec{u}$ whose projection along $\vec{w}$ is given by θ.

$\theta = 0$



$\theta \neq 0$

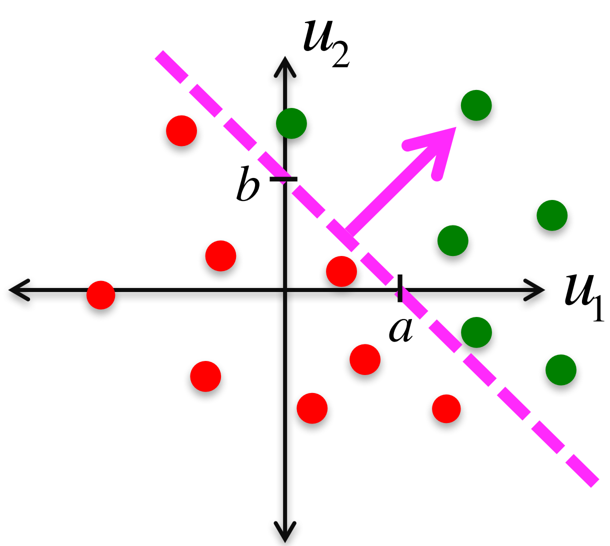


Classification in two dimensions

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

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…

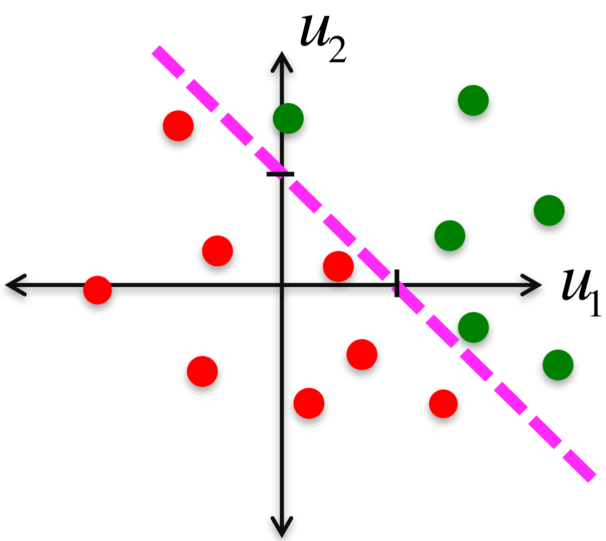
$$\vec{u}_a = (a \, , 0) \qquad \vec{u}_a \cdot \vec{w} = \theta$$

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

Learning classification in higher dimensions

- In two dimensions, you can basically look at the data and decide where the decision boundary should be.

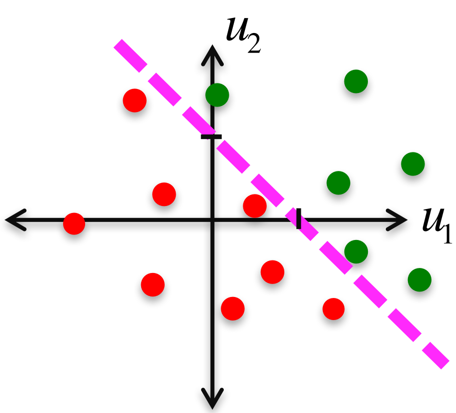
- But in higher dimensions this is a hard problem.

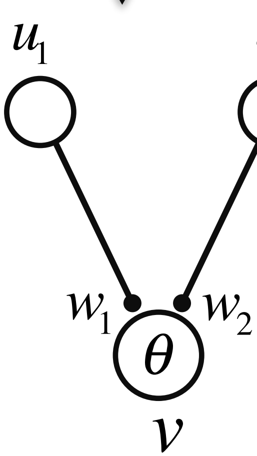


Perceptron learning

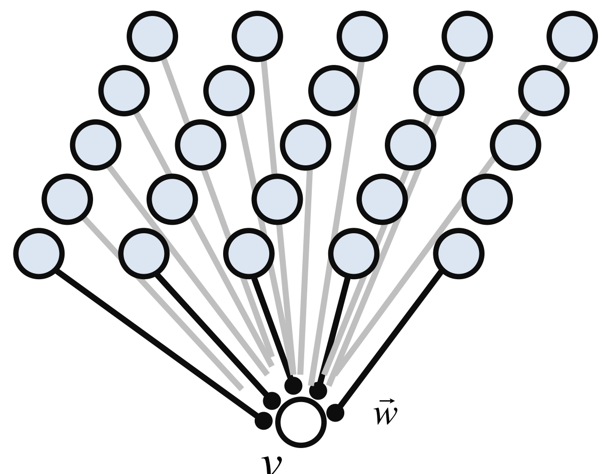
- How would we find the weight vector w that separates images of dogs from images of cats?

Low-dimensional





High-dimensional



Perceptron learning rule

Generic images sources unknown. All rights reserved. This content is excluded from our Creative Commons license. For more information, see https://ocw.mit.edu/help/faq-fair-use/.

Learning Objectives for Lecture 14

- Derive a mathematically tractable model of neural networks (the rate model)

- Building receptive fields with neural networks

- Vector notation and vector algebra

- Neural networks for classification

- Perceptrons

MIT OpenCourseWare
https://ocw.mit.edu/

9.40 Introduction to Neural Computation
Spring 2018

For information about citing these materials or our Terms of Use, visit: https://ocw.mit.edu/terms.