Other Applications in CS

CS 355: Introduction to Graphics and Image Processing

Math Applications in CS

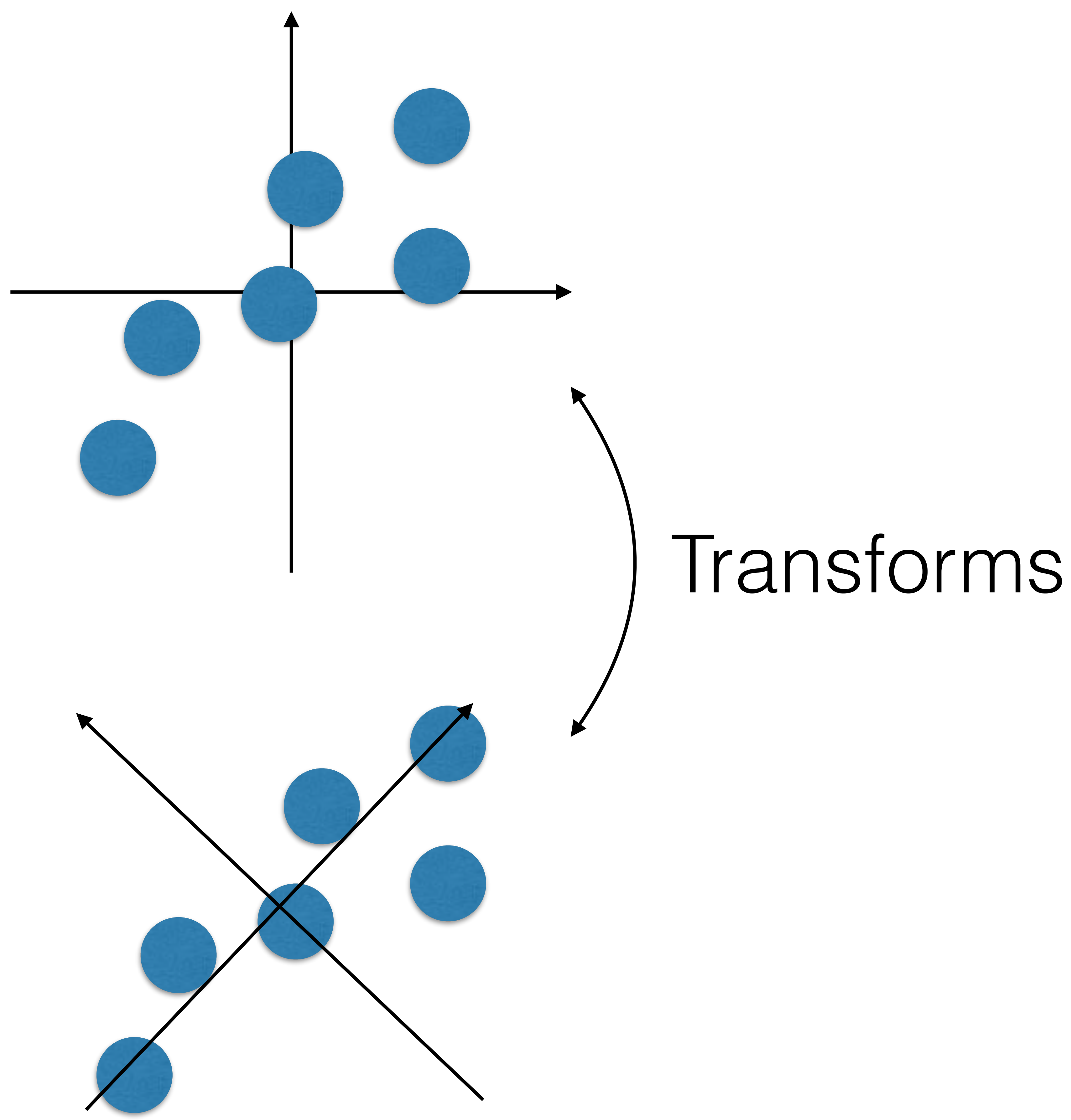
- All the “discrete math” from CS 235, 252, etc.
  - Sets, relations, functions, …
- Linear algebra
  - Geometric transformations
  - Data transforms
  - Systems of equations
  - Eigensystems

Data as Vectors

Lots of things can be thought of as points/vectors in some space

Data Transforms

- Common pattern in data analysis:
  - Represent data as vectors
  - Convert to a different coordinate system
  - Analyze (or change!) while in that coordinate system
  - Convert back (if needed)



Data Transforms

- Same form as any other change of coordinates
  - Transform using dot products
  - Convert back using weighted sum

transformed data

original data

basis set

$$g[i] = \sum_k f[k] \; e_i[k]$$

$$f[k] = \sum_i g[i] \; e_i[k]$$

Fourier Analysis

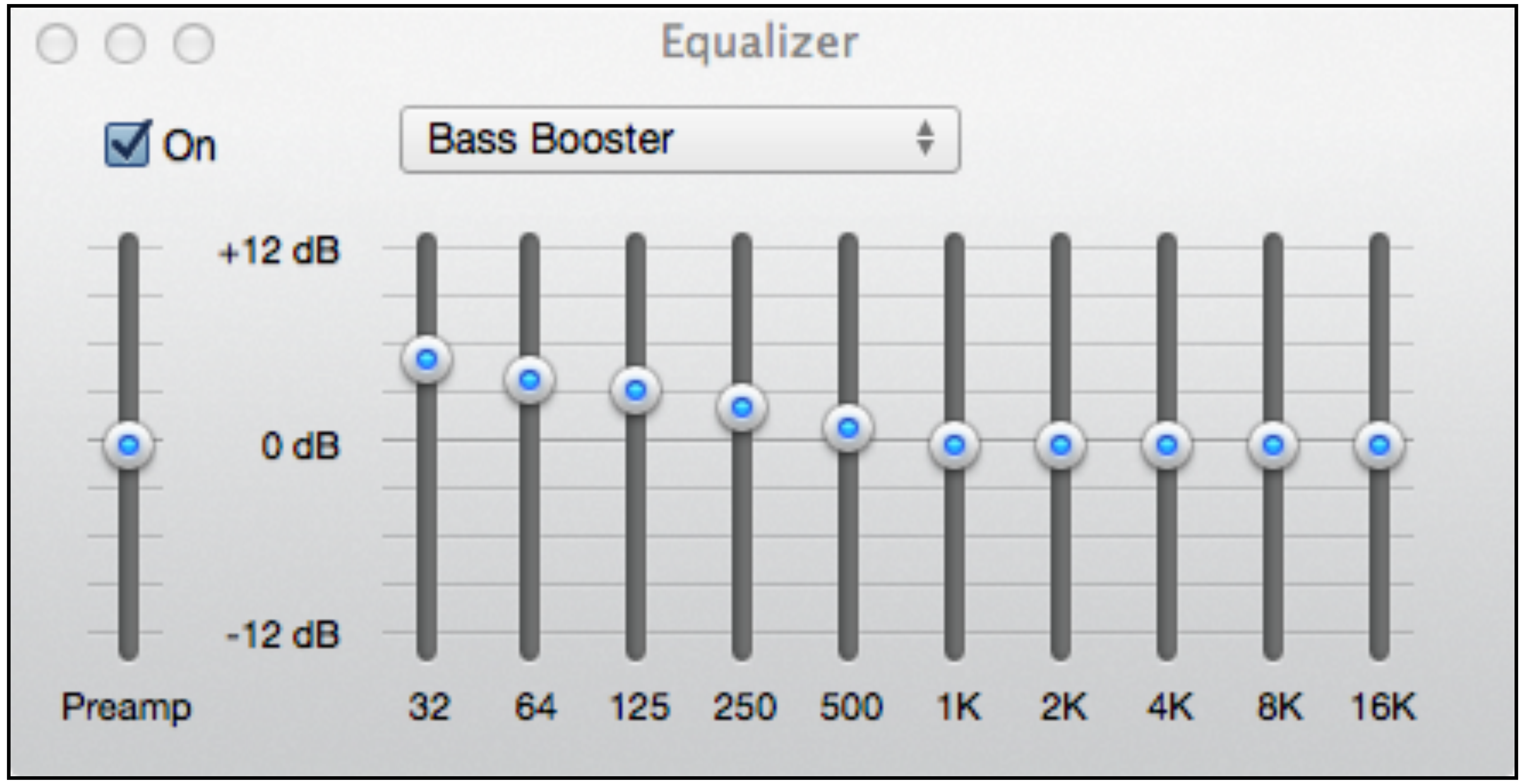
- Sampled sines and cosines of different frequencies form an orthonormal basis set
- Can decompose any waveform into a weighted sum of sines and cosines of different frequencies
- Great for analysis, manipulation, etc.

$$c[u] = \frac{1}{N} \sum_{u=0}^{N-1} f[t] \ \cos(2\pi ut/N)$$

$$s[u] = \frac{1}{N} \sum_{u=0}^{N-1} f[t] \ \sin(2\pi ut/N)$$

Frequency Manipulation

- Can use frequency-based representation to do manipulation
    - Boost bass/treble
    - Boost typical range of human speaking (hearing aides do this)
    - Suppress unwanted sounds
    - Smoothing / sharpening
    - And lots more…



Audio Compression

- Fact: your ear doesn’t hear all frequencies equally well
  (and it’s different for everybody)

- Idea: don’t not spend as many bits of precision on the ones we don’t
  as hear well anyway

Audio Compression

- Compression:
  - Convert to a frequency-based representation
  - Use more bits to store the coefficients for the frequencies we hear better; fewer for the ones we don’t hear well
  - Store in this form
- Decompression:
  - Use lossy coefficients and convert back to a PCM representation
  - Play!

This is how MP3 compression works!

Image Compression

- Can we do something similar with images?
- Fact: your eye is less accurate in sensing very rapid changes in brightness across an image (fine texture)
- Idea: use the same approach in the 2D frequency domain for images
- This is the basis of JPEG
  (uses Discrete Cosine Transform instead of Fourier)

Classification

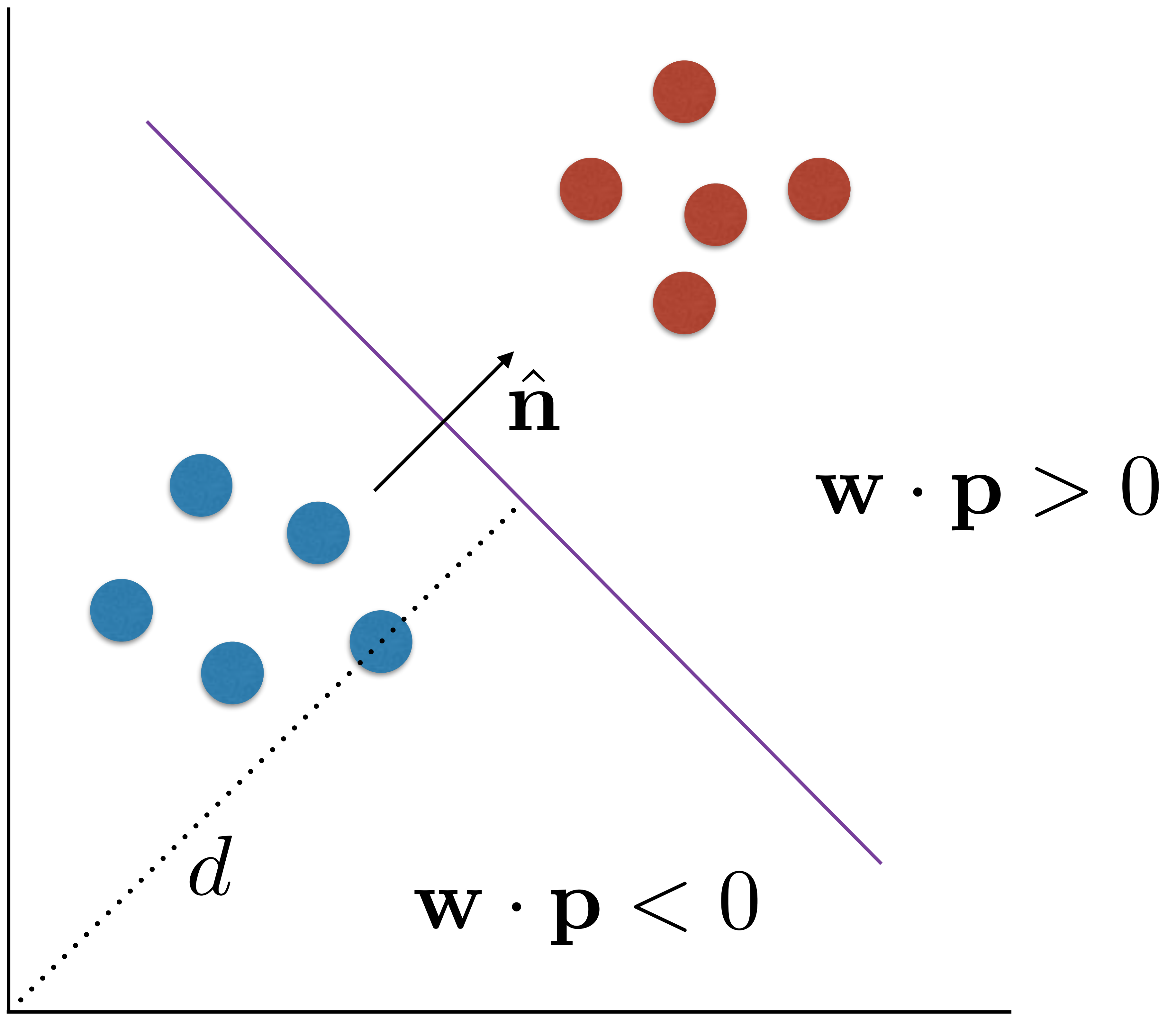
- Simple problem:
  - Two classes of things, with lots of examples of each
  - New thing — what kind is it?
- Approach:
  - Measure “features” of the things
  - Put features together in a vector
  - Look at the problem geometrically
  - Changing the coordinate system can make a huge difference!

(basis for pattern recognition, machine learning, other AI, …)

Classification

$$\mathbf{p} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \\ 1 \end{bmatrix}$$

$$\mathbf{w} = \begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ n_k \\ -d \end{bmatrix}$$



Classification

- But what if it's complicated, and you can't easily solve for **w**?
- Can you iteratively tweak the values in **w** until you "get it right"?
- The entries of **w** act as "weights" in a weighted combination of features, so it's called a *weight vector*

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \\ w_{k+1} \end{bmatrix} \quad \mathbf{p} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \\ 1 \end{bmatrix}$$

$$\mathbf{w} \cdot \mathbf{p} = \sum_{i=1}^{k} w_i \; x_i + w_{k+1}$$

This is basically the heart of what neural networks do!

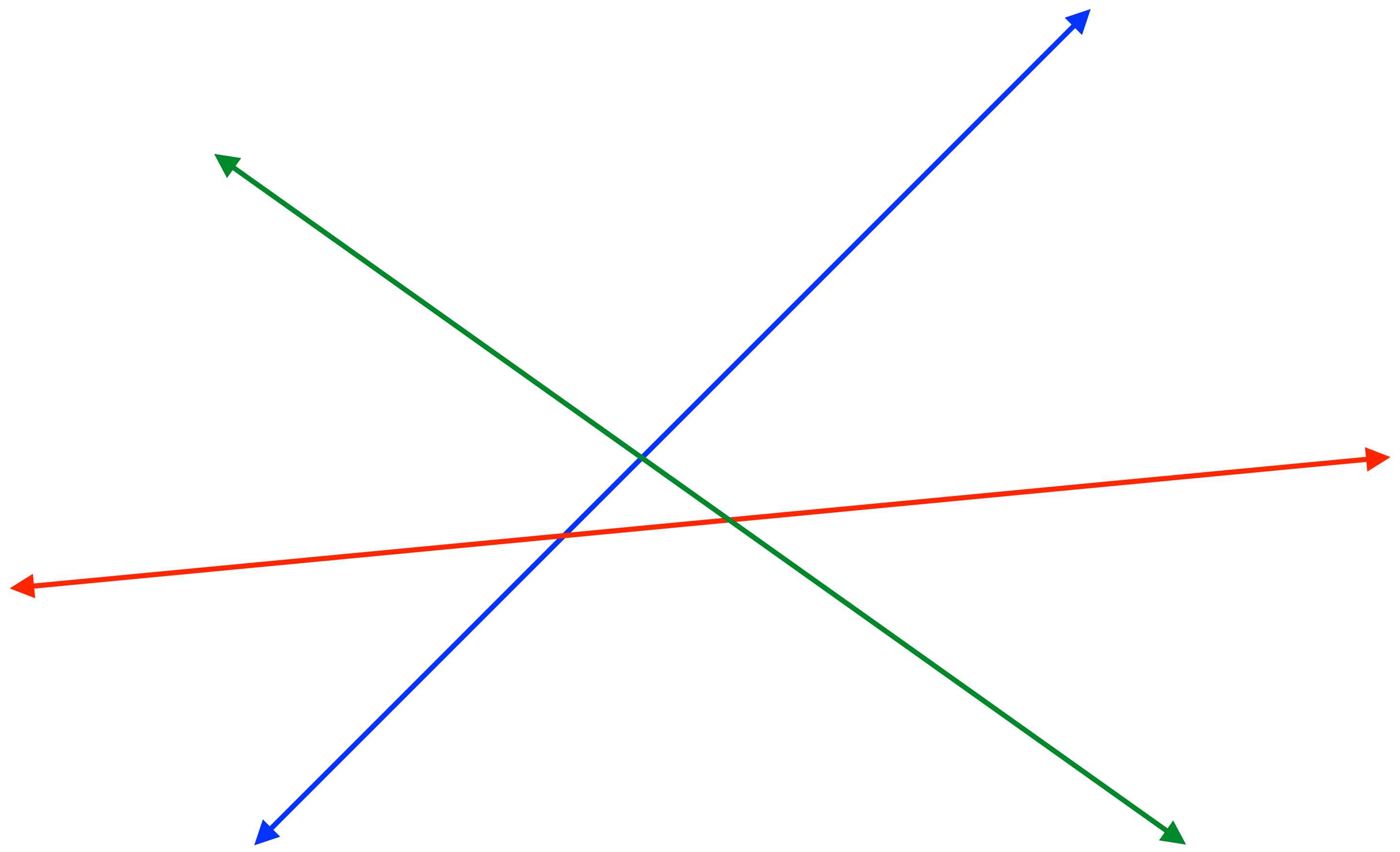
Systems of Equations

- If you need to solve for *n* unknowns, you need *n* equations, right?
- What if the data is noisy?
- Idea: get more data and let the noise "average out"
- But now there are too many equations!

$$\mathbf{Ax} = \mathbf{b}$$

Overconstrained Systems

- When you have too many equations, there may not be a solution
- Idea: get as close as possible to fitting all of the equations
- If you use a squared-error metric, this leads to a *least-squares solution*



$$\text{minimize} \quad \|\mathbf{Ax} - \mathbf{b}\|^2$$

Eigensystems

- The actions of some matrices are sometimes described more easily when converted to another coordinate system
- Some matrices are *pure scaling* along certain key directions
- A useful tool for analyzing these are *eigensystems*
  - Eigenvectors - directions of scaling
  - Eigenvalues - amount of scaling in each direction

Eigensystems

What does this matrix do?

$$\begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$$

Scales by 4 in the direction [1,1]
and by 2 in the direction [-1,1]

Eigenvectors:

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Eigenvalues:

$$4, 2$$

Coming up...

- Applications in Computer Vision (and CS 450) — Dr. Farrell
- Applications in Computer Graphics (and CS 455) — Dr. Egbert