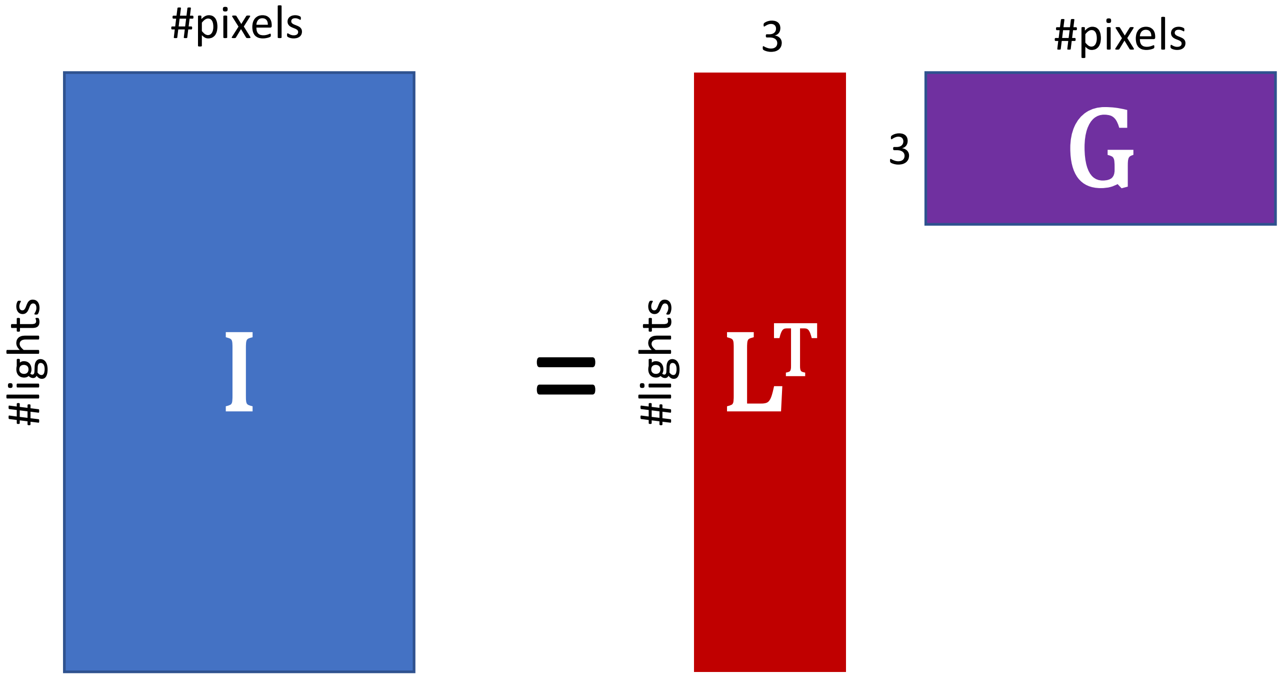
Photometric stereo

Multiple pixels: matrix form

$$\mathbf{I} = \mathbf{L}^T \mathbf{G}$$



Unknown Lighting

- What we've seen so far: [Woodham 1980]

- Next up: Unknown light directions [Hayakawa 1994]

Unknown Lighting

Surface normals Light directions

$$I = kN \cdot \ell L$$

Diffuse albedo

Light intensity

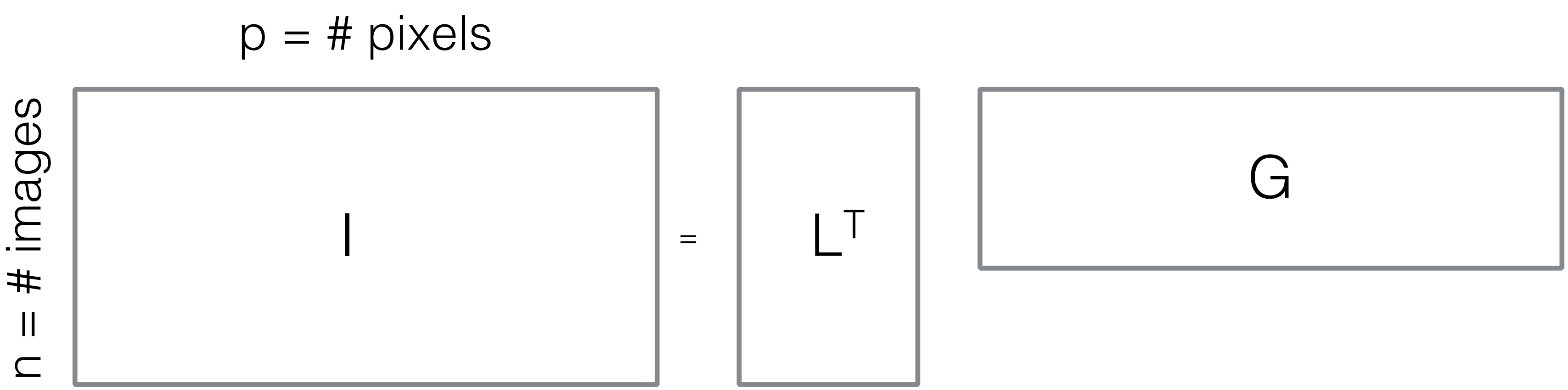
Unknown Lighting

Surface normals, scaled by albedo

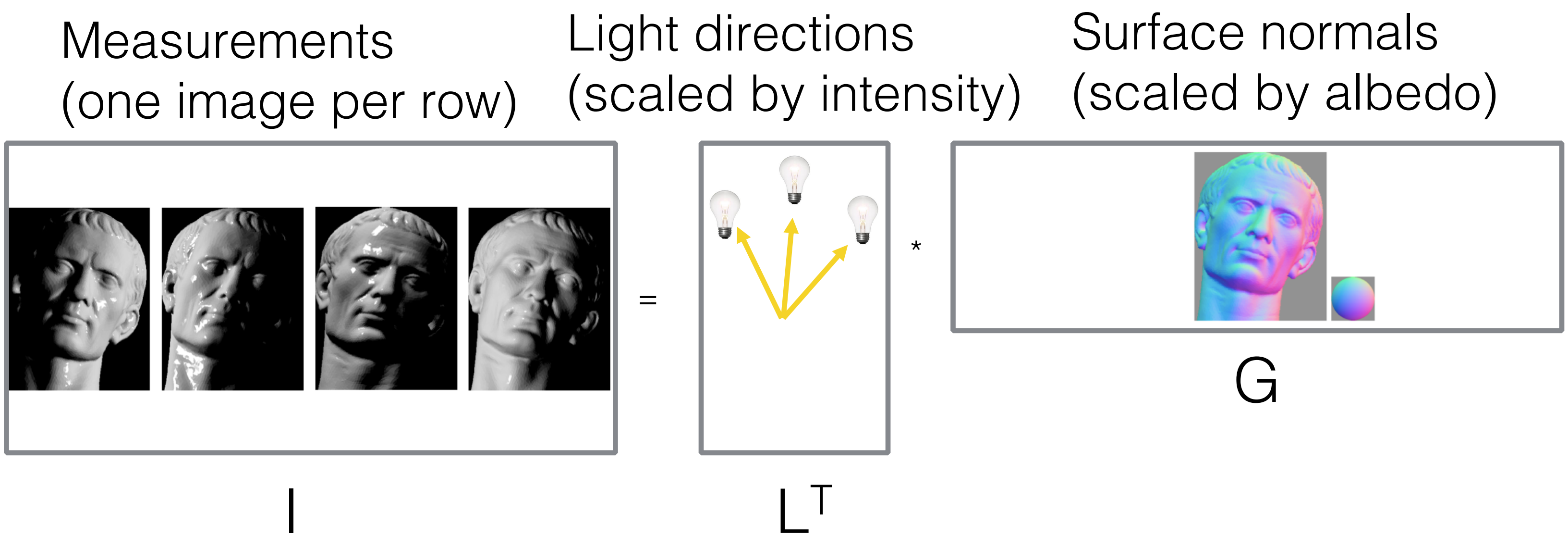
Light directions, scaled by intensity

$$I = N \cdot L$$

Unknown Lighting

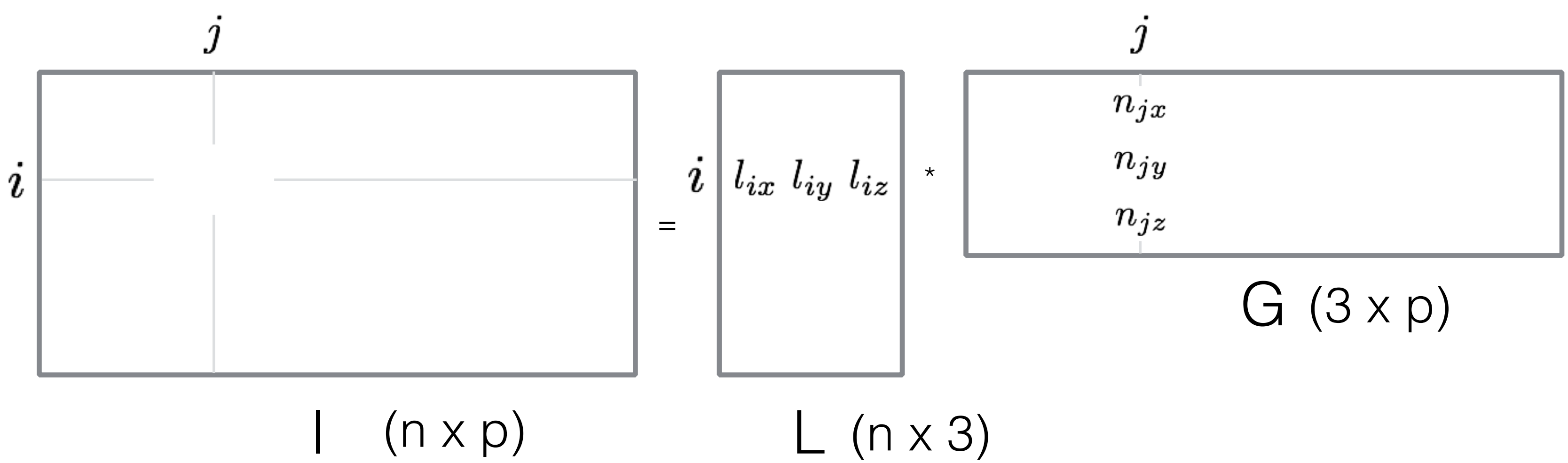


Unknown Lighting



Both L and G are now unknown!
This is a matrix factorization problem.

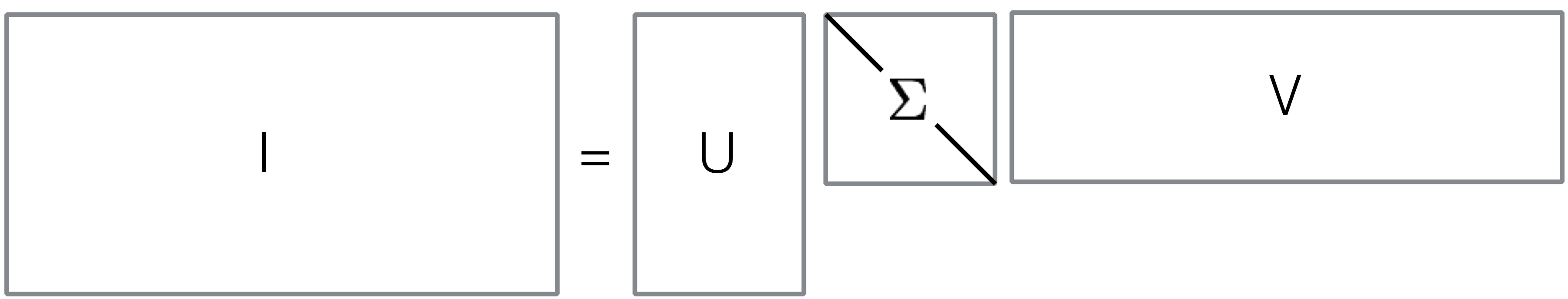
Unknown Lighting



There's hope: We know that I is rank 3

Unknown Lighting

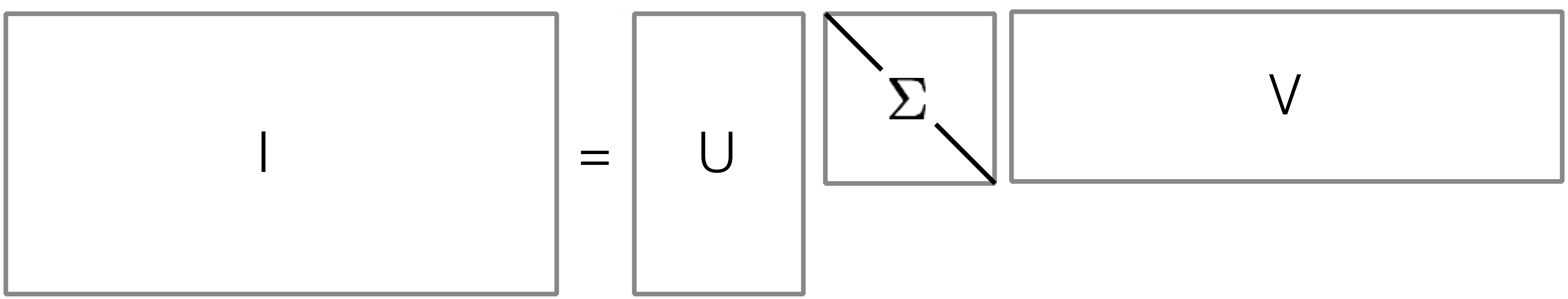
Use the SVD to decompose I:



SVD gives the best rank-3 approximation of a matrix.

Unknown Lighting

Use the SVD to decompose I:



SVD gives the best rank-3 approximation of a matrix.

What do we do with Σ?

Unknown Lighting

Use the SVD to decompose I:

$$I = U\sqrt{\Sigma}\;\sqrt{\Sigma}V$$

Can we just do that?

Unknown Lighting

Use the SVD to decompose I:

$$I = U\sqrt{\Sigma} \; A \; A^{-1} \; \sqrt{\Sigma}V$$

Can we just do that? …almost.

The decomposition is unique up to an invertible 3x3 A.

Unknown Lighting

Use the SVD to decompose I:

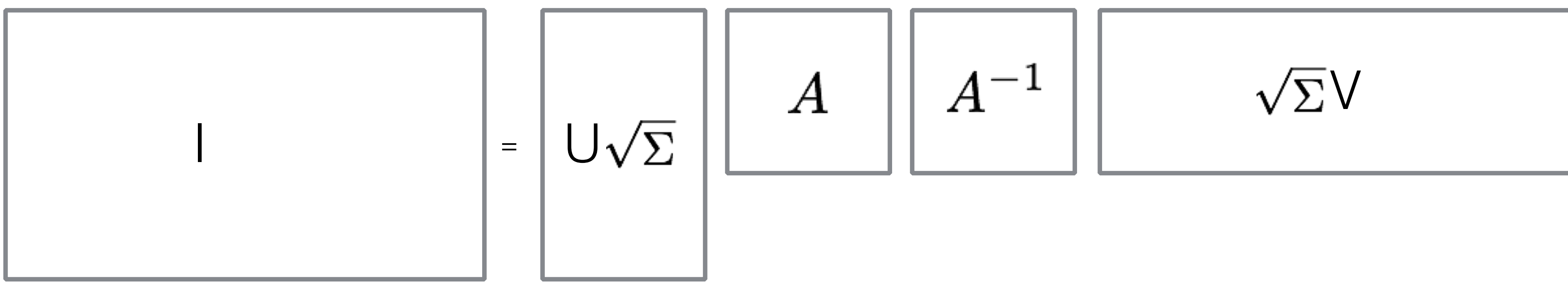
$$I = U\sqrt{\Sigma} \quad A \quad A^{-1} \quad \sqrt{\Sigma}V$$

Can we just do that? …almost. $L = U\sqrt{\Sigma}A, G = A^{-1}\sqrt{\Sigma}V$

The decomposition is unique up to an invertible 3x3 A.

Unknown Lighting

Use the SVD to decompose I:



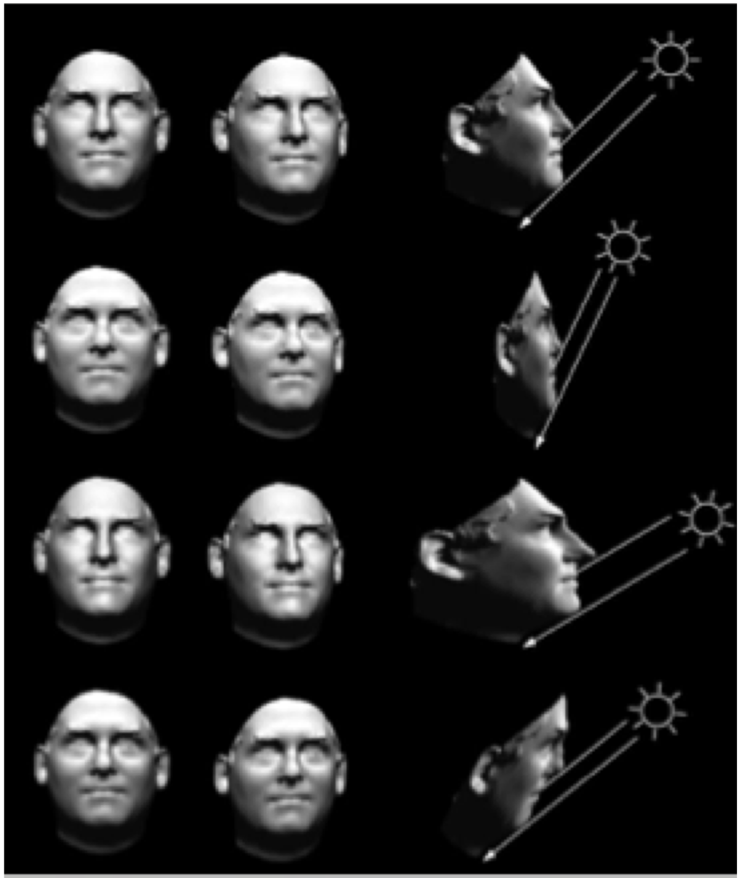
$$L = U\sqrt{\Sigma}A, G = A^{-1}\sqrt{\Sigma}V$$

You can find A if you know

- 6 points with the same reflectance, or
- 6 lights with the same intensity.

Unknown Lighting: Ambiguities

- Multiple combinations of lighting and geometry can produce the same sets of images.
- Add assumptions or prior knowledge about geometry or lighting, etc. to limit the ambiguity.



[Belhumeur et al.'97]

Recognition

Image classification

- Given an image, produce a label
- Label can be:
  - 0/1 or yes/no: *Binary classification*
  - one-of-k: *Multiclass classification*
  - 0/1 for each of k concepts: *Multilabel classification*

Image classification - Binary classification

Is this a dog?
Yes

Image classification - Multiclass classification

Which of these is it:
dog, cat or zebra?
Dog

Image classification - Multilabel classification

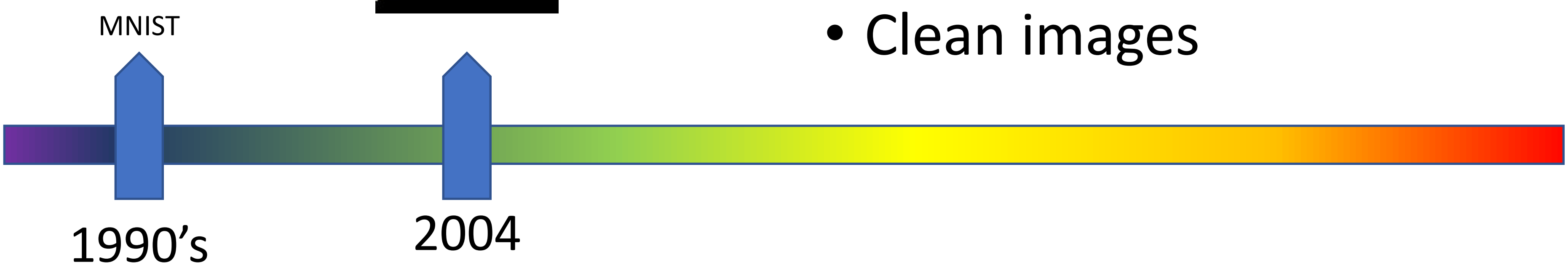
Is this a dog? Yes
Is this furry? Yes
Is this sitting down? Yes

A history of classification : MNIST

- 2D
- 10 classes
- 6000 examples per class

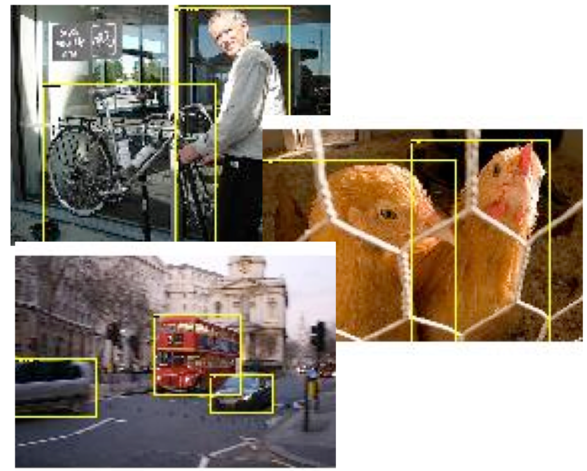
A history of classification : Caltech 101

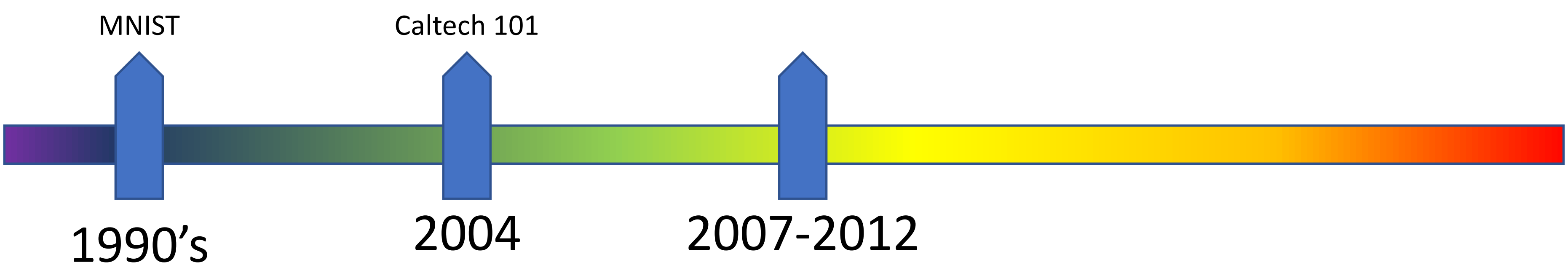
- 101 classes
- 10 classes
- 30 examples per class
- Strong category-specific biases
- Clean images



A history of classification: PASCAL VOC

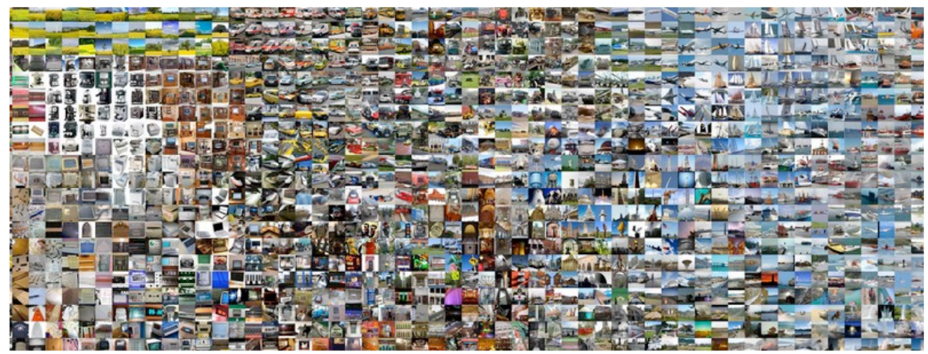
- 20 classes
- ~500 examples per class
- Clutter, occlusion, natural scenes

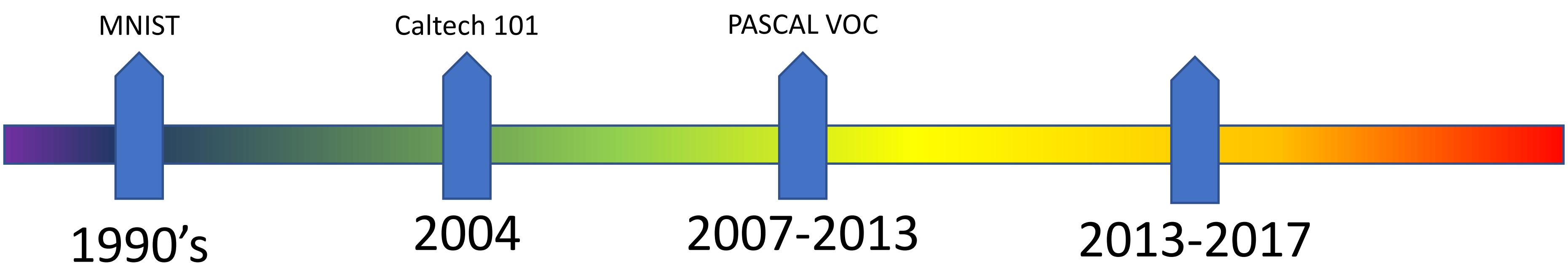




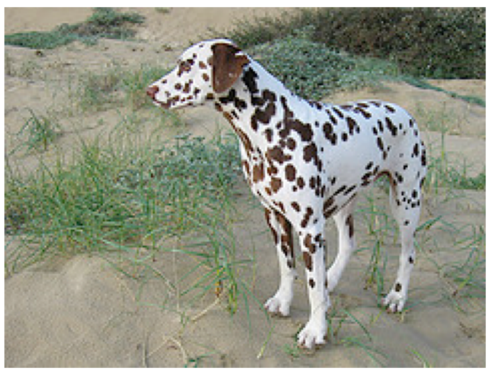
A history of classification: ImageNet

- 1000 classes
- ~1000 examples per class
- Mix of cluttered and clean images



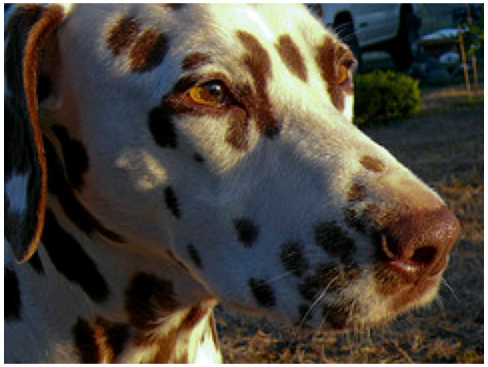


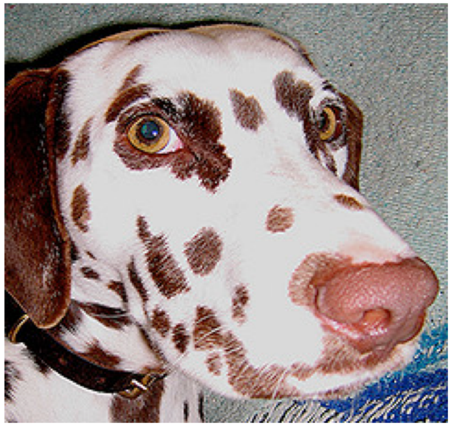
Why is recognition hard?



Pose variation

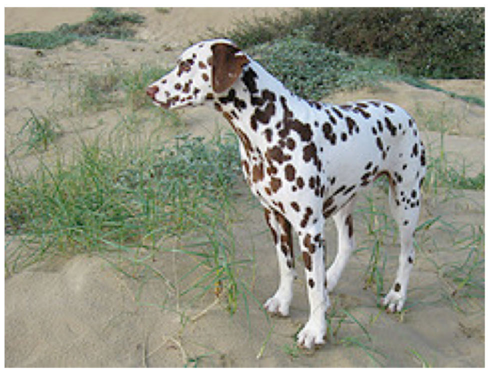
Why is recognition hard?





Lighting variation

Why is recognition hard?

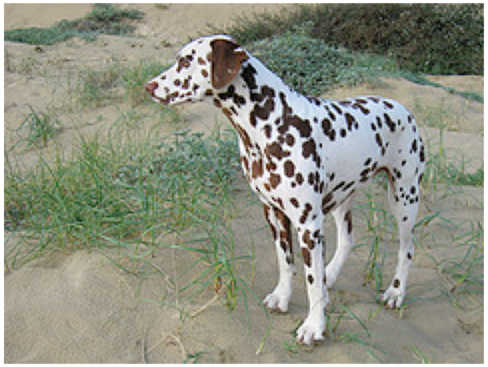


Scale variation

Why is recognition hard?

Clutter and occlusion

Why is recognition hard?



Intrinsic intra-class variation

Why is recognition hard?

Inter-class similarity

The language of recognition

- Boundaries of classes are often fuzzy
- “A dog is an animal with four legs, a tail and a snout”
- Really?

The language of recognition

- "… Practically anything can happen in an image and furthermore practically everything did" - Marr
- Much better to talk in terms of *probabilities*

$$\mathcal{X} \text{ :Images}$$
$$\mathcal{Y} \text{ :Labels}$$
$$\mathcal{D} \text{ :Distribution over } \mathcal{X} \times \mathcal{Y}$$

- *Joint distribution of images and labels* : P(x,y)
- *Conditional distribution of labels given image* : P(y|x)

Learning

- We are interested in the conditional distribution $P(y|x)$
- Key idea: teach computer visual concepts by *providing examples*

$\mathcal{X}$:Images

$\mathcal{Y}$:Labels

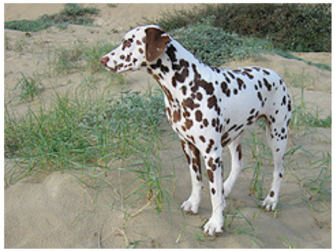
$\mathcal{D}$:Distribution over $\mathcal{X} \times \mathcal{Y}$

Training Set

$$S = \{(x_i, y_i) \sim \mathcal{D}, i = 1, \ldots, n\}$$

Example

- Binary classifier “Dog” or ”not Dog”
- Labels: {0, 1}
- Training set

{( , 1), ( , 1), ( , 0) , … }

Choosing a model class

- Will try and find P(y = 1 | x)
- P(y=0 | x) = 1 - P(y=1 | x)
- Need to find $h : \mathcal{X} \rightarrow [0, 1]$
- But: *enormous number of possible mappings*

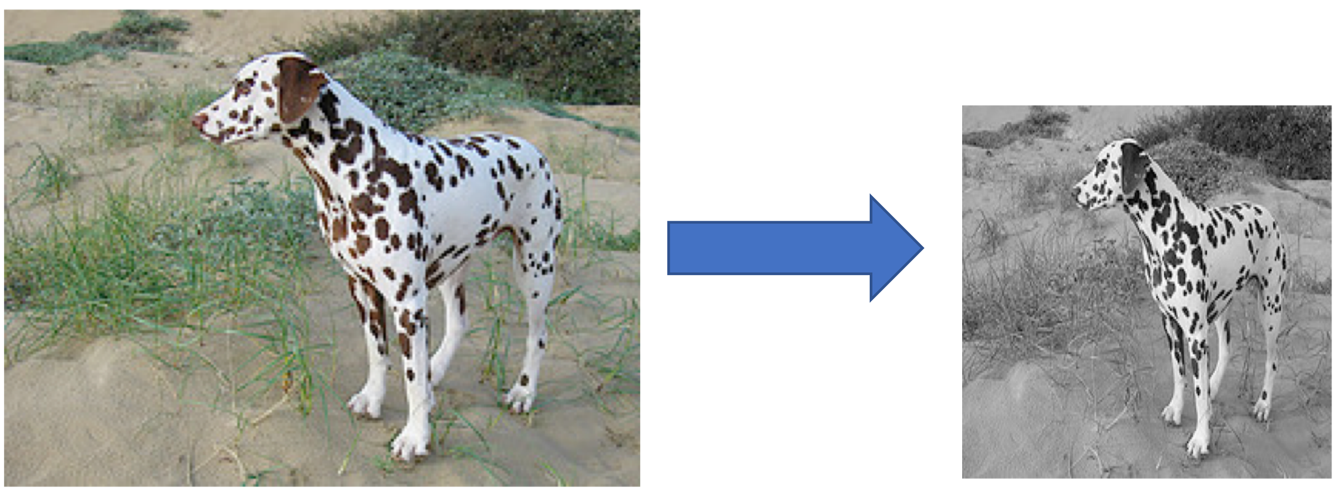
Choosing a model class

$$h : \mathcal{X} \rightarrow [0, 1]$$

- Assume h is a linear classifier in feature space
- Feature space?
- Linear classifier?

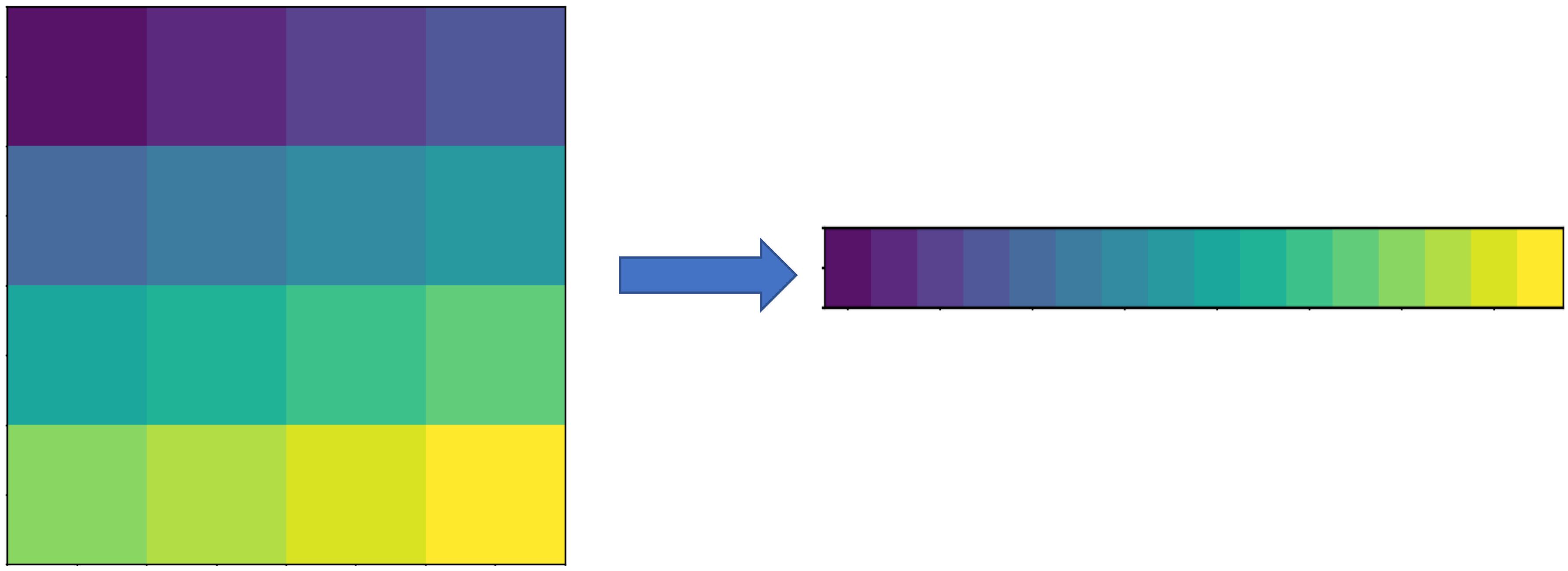
Feature space: representing images as vectors

- Represent an image as a vector in $\mathbb{R}^d$
- Simple way: step 1: convert image to gray-scale and resize to fixed size



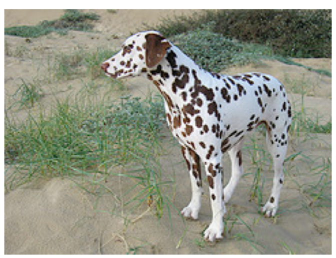
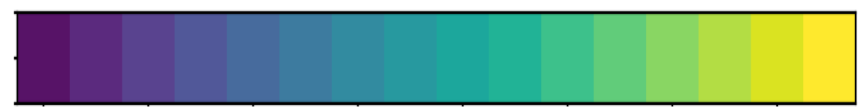
Feature space: representing images as vectors

- Step 2: Flatten 2D array into 1D vector



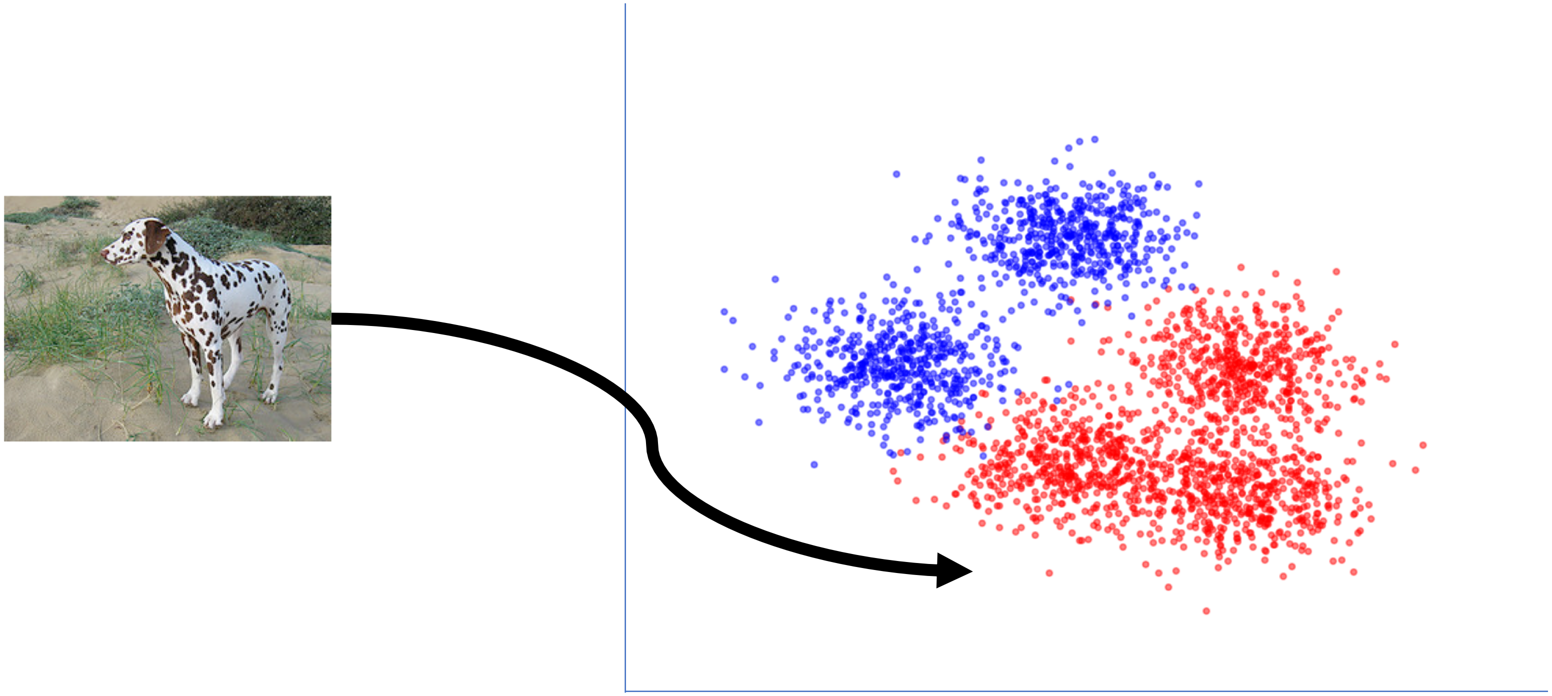
Feature space: representing images as vectors

- Can represent this as a *function* that takes an image and converts into a vector

$\phi\,($  $)$ $=$ 

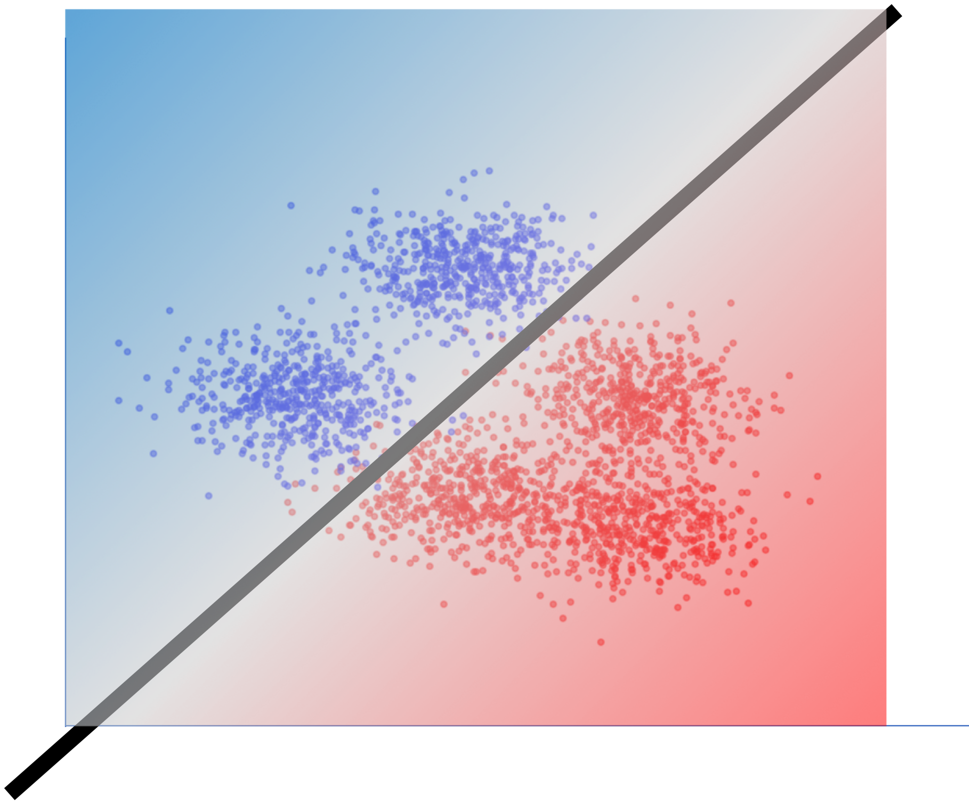
Linear classifiers

- Given an image, can use ϕ to get a vector and plot it as a point in high dimensional space



Linear classifiers

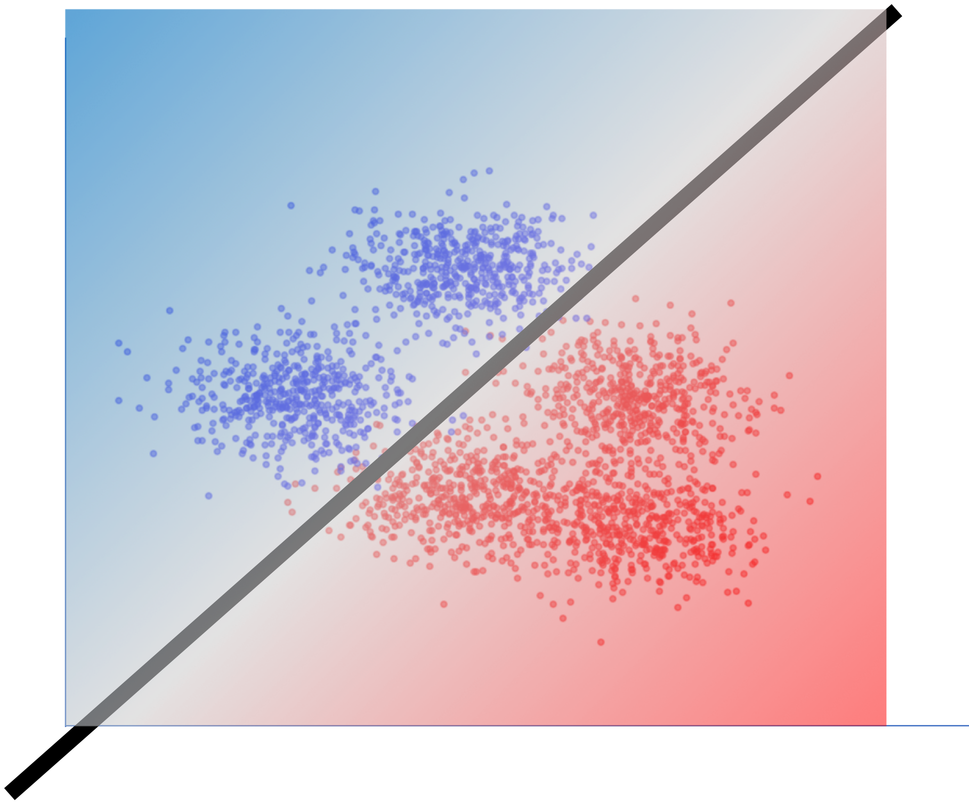
- A linear classifier corresponds to a hyperplane
  - Equivalent of a line in high-dimensional space
  - Equation: $w^T x + b = 0$
- Points on the same side are the same class



Linear classifiers

- p(y = 1 | x) is high on the red side and low on the blue side
- A common way of defining p:

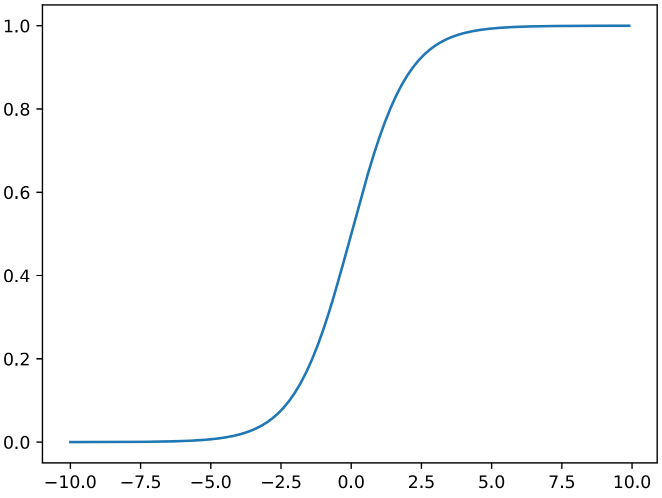
$$p(y = 1\,|x)$$
$$= \sigma(w^T x + b)$$
$$= \underbrace{\frac{1}{1 + e^{-(w^T x + b)}}}_{\text{sigmoid function}}$$



Linear classifiers in feature space

$$h(x; \mathbf{w}, b) = \sigma(\mathbf{w}^T \phi(x) + b)$$

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



Linear classifiers in feature space

$$h(x; \mathbf{w}, b) = \sigma(\mathbf{w}^T \phi(x) + b)$$

- *Family* of functions depending on **w** and b
- Each function is called a *hypothesis*
- Family is called a *hypothesis class*
- Hypotheses indexed by **w** and b
- Need to find the best hypothesis = need to find best **w** and b
- **w** and b are called *parameters*

Training: Choosing the best hypothesis

- Use training set to find *best-fitting* hypothesis

$$S = \{(x_i, y_i): i = 1, \dots, n\}$$

- Question: how do we define fit?

Training: Choosing the best hypothesis

- Use training set to find *best-fitting* hypothesis
- Question: how do we define fit?
- Given (x,y), and candidate hypothesis $h(\cdot; \mathbf{w}, b)$
  - $h(x; \mathbf{w}, b)$ is estimated probability label is 1
  - Idea: compute estimated probability for true label y
  - Want this probability to be high
  - *Likelihood*

$$li(h(x;\mathbf{w},b),y) = \begin{cases} h(x;\mathbf{w},b) & \text{if } y = 1 \\ 1 - h(x;\mathbf{w},b) & \text{ow} \end{cases}$$

An alternate expression for the hypothesis

$$li(h(x;\mathbf{w},b),y) = \begin{cases} h(x;\mathbf{w},b) & \text{if } y = 1 \\ 1 - h(x;\mathbf{w},b) & \text{ow} \end{cases}$$

An alternate expression for the hypothesis

$$li(h(x;\mathbf{w},b),y) = \begin{cases} h(x;\mathbf{w},b) & \text{if } y = 1 \\ 1 - h(x;\mathbf{w},b) & \text{ow} \end{cases}$$

$$li(h(x;\mathbf{w},b),y) = h(x;\mathbf{w},b)^{y}(1 - h(x;\mathbf{w},b))^{(1-y)}$$

Training: Choosing the best hypothesis

$$li(h_{\mathbf{w}}(x), y) = h_{\mathbf{w}}(x)^y (1 - h_{\mathbf{w}}(x))^{1-y}$$

- Likelihood of a single data point
- Fit = *total likelihood of entire training dataset*

$$S = \{(x_i, y_i) \sim \mathcal{D}, i = 1, \ldots, n\}$$

$$\prod_{i=1}^{n} h(x_i; \mathbf{w}, b)^{y_i} (1 - h(x_i; \mathbf{w}, b))^{(1-y_i)}$$

Training: Choosing the best hypothesis

$$\prod_{i=1}^{n} h(x_i; \mathbf{w}, b)^{y_i} (1 - h(x_i; \mathbf{w}, b))^{(1-y_i)}$$

- Use log likelihood

$$lli(\mathbf{w}, b) = \sum_{i=1}^{n} y_i \log h(x_i; \mathbf{w}, b) + (1 - y_i) \log(1 - h(x_i; \mathbf{w}, b))$$

- Pick the hypothesis that maximizes log likelihood
  - Each hypothesis corresponds to a setting of **w** and b
  - *Maximization problem*

$$\max_{\mathbf{w}, b} \sum_{i=1}^{n} y_i \log h(x_i; \mathbf{w}, b) + (1 - y_i) \log(1 - h(x_i; \mathbf{w}, b))$$

Training: Choosing the best hypothesis

- Maximizing log likelihood = *Minimizing negative log likelihood*

$$\max_{\mathbf{w},b} \sum_{i=1}^{n} y_i \log h(x_i;\mathbf{w},b) + (1-y_i)\log(1-h(x_i;\mathbf{w},b))$$

$$\equiv \min_{\mathbf{w},b} -(\sum_{i=1}^{n} y_i \log h(x_i;\mathbf{w},b) + (1-y_i)\log(1-h(x_i;\mathbf{w},b)))$$

Training: Choosing the best hypothesis

- Negative log likelihood is a *loss function*

$$L(h(x;\mathbf{w},b),y) = -y\log h(x;\mathbf{w},b) + (1-y)\log(1 -$$

- *Training = minimizing total loss on a training set*

$$\min_{\mathbf{w},b}\sum_{i=1}^{N} L(h(x_i;\mathbf{w},b),y_i)$$

General recipe

- Fix hypothesis class
  $$h_{\mathbf{w}}(x) = \sigma(\mathbf{w}^T \phi(x))$$
- Define loss function
  $$L(h_{\mathbf{w}}(x), y) = (-y \log h_{\mathbf{w}}(x) + (1 - y) \log(1 - h_{\mathbf{w}}(x)))$$
- Minimize total loss on the training set

$$\min_{\mathbf{w}} \sum_{i=1}^{n} L(h_{\mathbf{w}}(x_i), y_i)$$

- *Why should this work?*
- *How do we do the minimization in practice*