Histogram of Oriented Gradient (HOG) Features:

feature descriptor used in computer vision & Image Processing for the purpose of object detection

- counts occurrences of gradient orientation in localized portions of an image

Algorithm:

1. compute the gradient values with the following filtering Kernels for a derivative mask in the horozontal and vertical direction
   a) $D_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$ and $D_x = [-1 \ 0 \ 1]$
   b) $I_x = I * D_x$ and $I_y = I * D_y$
   c) Magnitude of gradient: $|G| = \sqrt{I_x^2 + D_x^2}$
   d) Orientation of Gradient: $\Theta = \tan^{-1} \frac{I_y}{I_x}$
2. Divide the Image into small connected regions called cells and each pixel within a cell casts a weighted vote for an orientation-based histogram based on the values in the gradient computation
   a) cells are rectangular and histogram channels are spread over 0-180 degrees because "unsigned" gradients
   b) weighted vote = gradient magnitude
3. The local histograms from the cells can be contrast-normalized by calculating a measure of intensity across several cells called block
   b) HOG descriptor is the vector of the components of the normalized cell histograms from all block regions

Random Projection:

Reduce the dimensions of a $m \times n$ matrix to a $m \times K$ matrix by multiplying the original matrix by a random matrix of size $n \times K$.

- random matrix created by 1, 0, -1 entries
  - + 1/6 chance 1, 1/6 chance -1
  - + 2/3 chance 0

Example:

$$\begin{bmatrix} 2 & 3 & 4 \\ 5 & 6 & 7 \\ 2 & 3 & 4 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} -1 & 7 \\ -1 & 13 \\ -1 & 7 \end{bmatrix}$$

original matrix   random matrix   reduced dim

Bernoulli Projection: same as random projection except random matrix is 50/50 chance 1 or -1

Sorbel Masks:

Derivative mask used for Edge Detection in the vertical and horizontal directions

Vertical Mask

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Horizontal Mask

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

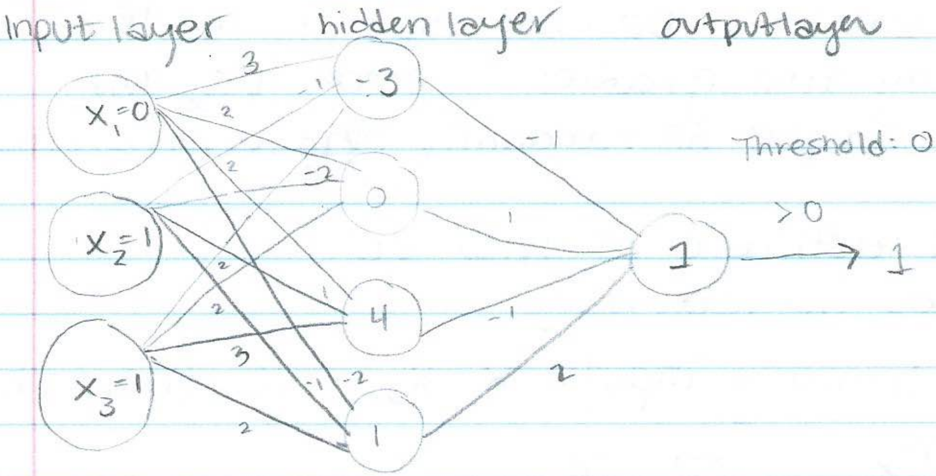
*2's for smoothing w/ respect to center point

Example:

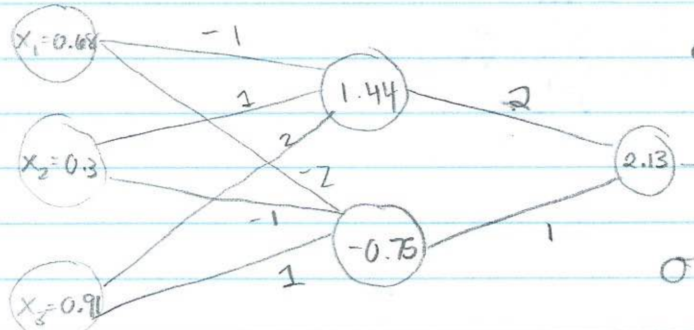
225	225	0	100	10	100	120	130	90
225	80	0	10	10	30	0	50	0
200	70	0	20	10	0	10	40	225
100	48	0	10	0	40	110	200	225
100	60	0	0	0	60	100	50	0
200	10	0	10	0	0	120	60	200
150	40	0	10	30	0	100	200	100
50	20	0	10	40	20	0	140	225
100	60	0	50	40	100	40	100	225

	335	245	180	190	310	410	125	
	189	32	10	10	-120	-380	-435	
	120	30	50	-20	-200	-250	115	
	-24	38	0	40	70	160	295	
	-10	10	-50	-60	90	-90	-400	
	130	-10	-40	-100	40	140	-65	
	10	60	-70	-220	-150	120	135	

Perceptron Example



Sigmoid Neurons



$\sigma(w \cdot x + b)$ where $\sigma(z) = \frac{1}{1+e}$

let $b = 0.56$

$\rightarrow 0.9364$

$\sigma(2.69) = \frac{1}{1+e^{-2.69}} = 0.93$

Stochastic gradient descent

cost function:

$$C(w,b) = \frac{1}{2n}\sum_x \|y(x) - a\|^2$$ good job if $C(w,b) \approx 0$

want to find a set of weights & biases to make $C(w,b)$ as small as possible

Problem:
Average cost over $C_x = \frac{\|y(x) - a\|^2}{2}$ for individual training examples, but this takes a long time and learning is slow

Stochastic Gradient Descent speeds up learning by computing the gradient vector ∇C_x for a small sample of randomly chosen training inputs

→ good estimate of true ∇C

→ speeds up learning

Random training inputs $X_1, X_2, \ldots X_m$ (mini-batch)

$$\frac{\sum_{j=1}^{m} \nabla C_{X_j}}{m} \approx \frac{\sum_x \nabla C_x}{n} = \nabla C$$

Then

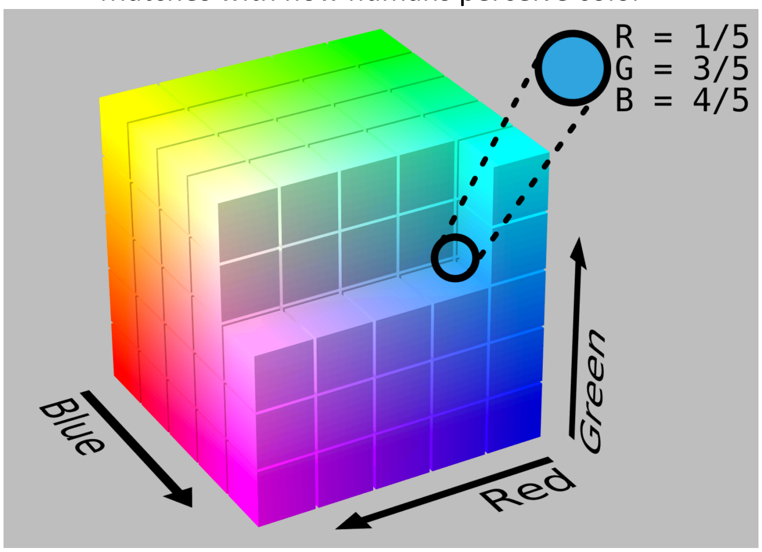
$$\nabla C = \frac{1}{m} \sum_{j=1}^{m} \nabla C_{X_j}$$

Color Spaces

The purpose of a color model is to facilitate the specification of colors in a standard, generally accepted way. Some color systems are better suited for hardware applications, however these color models are not practical for human interpretation of color. For example, humans describe color by hue, saturation and brightness.

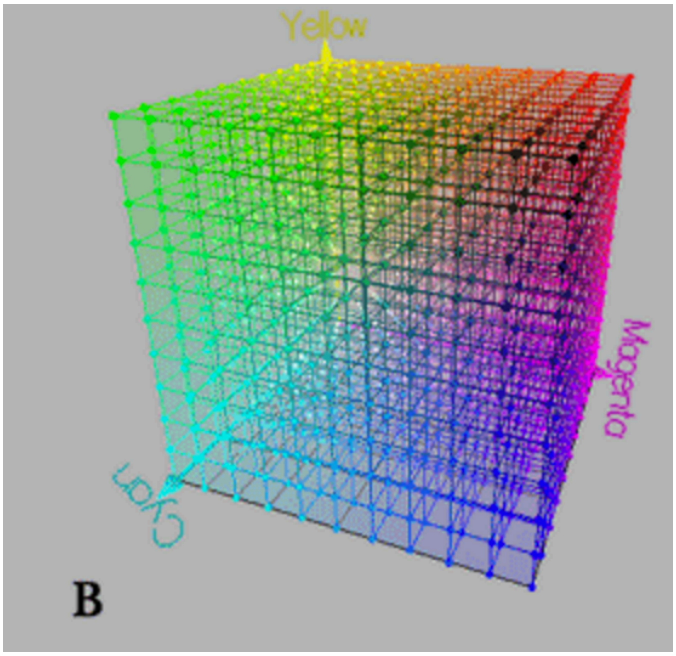
RGB: each color appears in its primary spectral components of red, green and blue and is based on Cartesian coordinates

- Matches with how humans perceive color

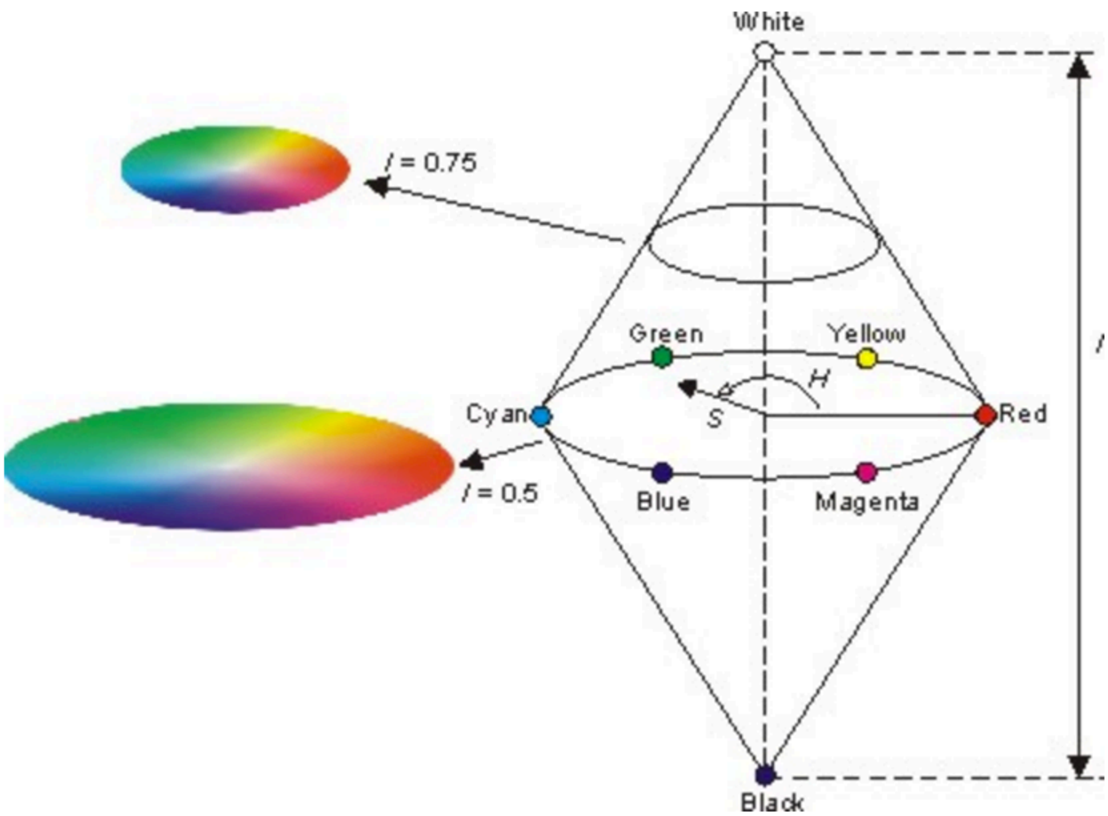


CMY: cyan, magenta, and yellow are secondary colors of light, or primary colors of pigment. Assuming the color values have been normalized to the range [0,1] (by dividing each by 255), the conversion from RGB to CMY is as follows

$$\begin{matrix} C \\ M \\ Y \end{matrix} = \begin{matrix} 1 \\ 1 \\ 1 \end{matrix} - \begin{matrix} R \\ G \\ B \end{matrix}$$



HSI: Hue, saturation, and intensity color model that is ideal for image processing algorithms and based on intuitive color descriptions. Conceptually, if you turn the RGB cube and stand it on the black axis and run a plane perpendicular to this axis we get the color points.



Color Space Comparisons

RGB:

Black (0,0,0)

White (255,255,255)

Red (255,0,0)

Green (0,255,0)

Blue (0,0,255)

Yellow (255,255,0)

CMY:

Black (0,0,0)

White (1,1,1) − (1,1,1) = (0,0,0)

Red (1,1,1) − (1,0,0) = (0,1,1)

Green (1,1,1) − (0,1,0) = (1,0,1)

Blue (1,1,1) − (0,0,1) = (0,0,1)

Yellow (1,1,1) − (1,1,0) = (1,0,0)

HSI:

$$H = \begin{cases} \theta & \text{if } B \leq G \\ 360 - \theta & \text{if } B > G \end{cases} \qquad \theta = \cos^{-1}\left\{ \frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-B)(G-B)]^{1/2}} \right\}$$

$$S = 1 - \frac{3}{(R+G+B)}[\min(R,G,B)]$$

$$I = \frac{1}{3}(R+G+B)$$

when R, G, B values normalized in Rang [0,1]

Black (0°, 0, 0)

White (0°, 0, 1)

Red (0°, 1, 0)

Green (120°, 1, 0)

Blue (240°, 1, 0)

Yellow (60°, 1, 1)