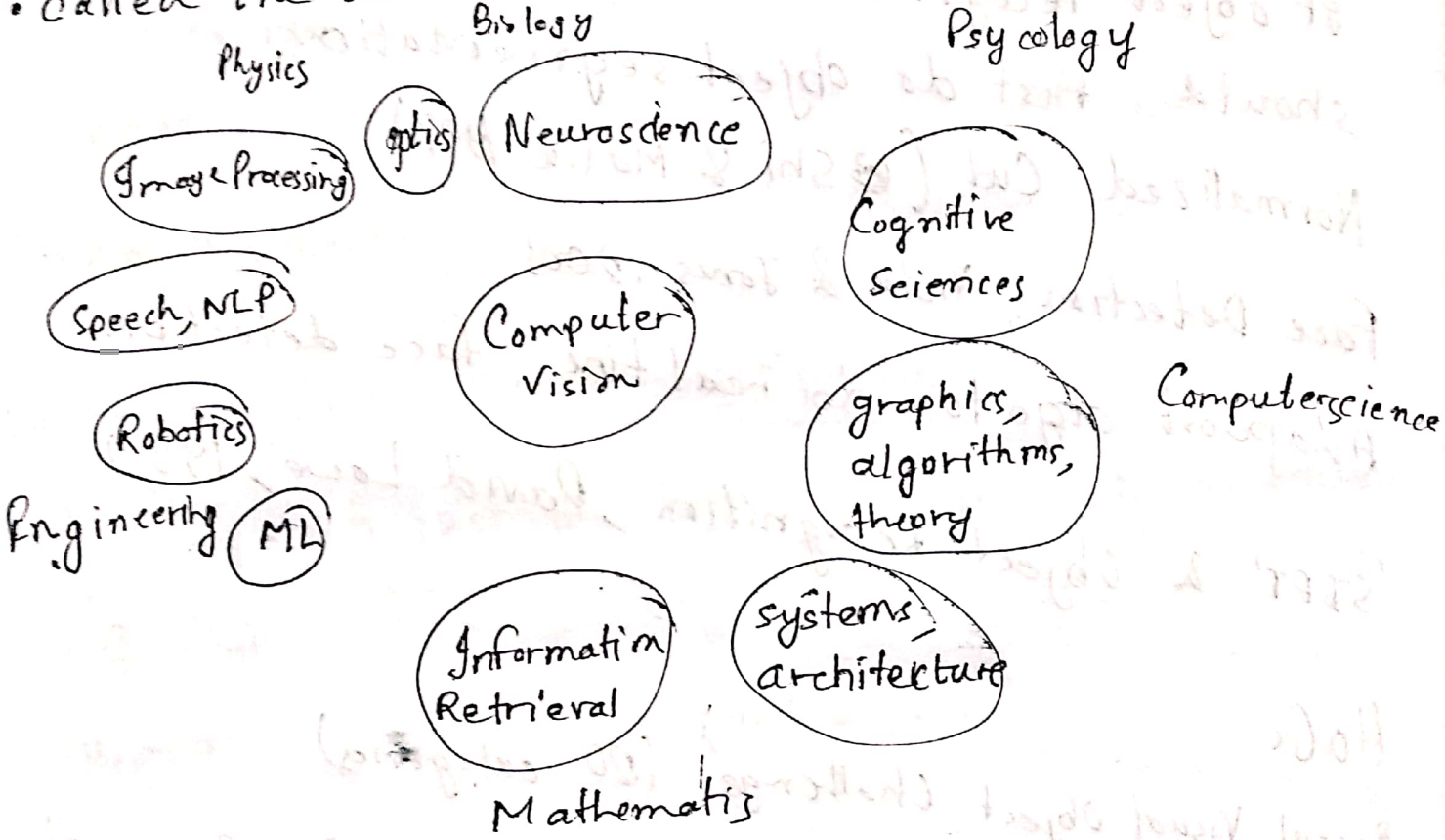
CS231n: Convolutional Nets for Visual Recognition

Fei-Fei Li; Justin Johnson; Serena Yueng

Computer vision is the study of visual data

- called the dark matter of internet



Hubel & Wiesel, 1959

Electrical signal from brain ⟷ mapping ⟷ stimulus

Block world [Larry Roberts, 1963]

1966 → Summer Vision Project

Generalized Cylinder
1979

Pictorial Structure
1973

Nonm.
If object recognition is too hard, may be we should first do object segmentation
Normalized Cut (Shi & Malik 1997)
Face Detection, Viola & Jones, 2001
Adaboost algo to do real time face detection
'SIFT' & Object Recognition, David Lowe, 1990

HoG
Pascal Visual Object Challenge (20 categories)
Imagenet (Deng, Dong, Socher, Li, & Fei-Fei, 2009)
Stanford & Princeton
(22K categories & 14M images)
Took 3 years (took help from Amazon Mechanical Turk)
Imagenet Classification Challenge

There is a number of visual recognition problems that are related to image classification, such as object detection, image captioning

Lecanet. al (conv-net)

Alexnet

Image → Captions

Lecture 2: Image Classification Pipeline

- An image is just a big grid of numbers between [0, 255]

Attempts: finding edges (not scalable)

Data Driven approach:

1) Collect data
2) Use ML to train
3) Evaluate the classifier

Distance Metric

L1 Distance ⇒ $d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$

L2 dist ⇒ $d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$

Nearest Neighbor $O(1)$

Memorize data

for each testimage
    find closest train image $O(N)$
    Predict label of nearest image

This is bad: we want classifiers that are fast at prediction; slow for training is ok

k-nearest neighbor (to get rid of noisy answer)

Hyperparameters: choices about the algo that we set rather than learn

Setting hyper params

K-NN on images never used.

- very slow at test time
- distance metric not informative (not good to check similarity)
- curse of dimensionality

Linear Classification

Parametric Approach

$f(x, W) \longrightarrow$ 10 numbers giving class scores

W ↑ Weights

$$f(x, W) = Wx + b$$

$f(x, W)$: 10×1 | W: 10×3072 | x: 3072×1 | b: 10×1

→ Linear classifier is learning only one template

Lecture 3 - Loss function & Optimization

Challenges of recognition

i) Viewpoint
ii) Illumination
iii) Deformation
iv) Occlusion
v) Clutter
vi) Intraclass variation

- Define a loss function that quantifies our unhappiness with the scores across the training
- Come up with a way of efficiently finding parameters that minimize the loss function

$$L = \frac{1}{N} \sum_i L_i(f(x_i, W), y_i)$$

Multiclass SVM loss

$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$

s_{y_i} : true value

$$= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \qquad s = f(x_i, W)$$

Then, $L = \frac{1}{N} \sum_{i=1}^{N} L_i$

What happen to loss if a score that's high is changed a bit?

→ Nothing happens because it still returns zero loss.

What is the min/max possible loss for SVM

→ min → 0
max → ∞

Q3| At initialization W is small so all $s \approx 0$.

What is the loss?

(number of classes - 1)

Q4| What if sum as ~~ad~~ over all classes?

The loss increases by 1

This is just for convention we omit the correct class so that our minimum loss is zero.

Q5| What if we used mean instead of sum?

→ answer would be same because we don't care about true scores

Q5| What if $\sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)^2$

→ This would end up ~~giving~~ to a different loss function that's not linear

Q6| If we find a W that gives $L=0$. Is this W unique?

No, there are many other Ws. Like $2W$

Regularization: Model should be simple so that it works on test data

$$L(W) = \frac{1}{N} \sum_{i=1}^{N} L_i(f(x_i, W), y_i) + \lambda R(W)$$

Occam's Razor:

Among competing hypotheses the simplest is the best

L2 Regularization $R(W) = \sum_k \sum_l W_{k,l}^2$

L1 Regularization $R(W) = \sum_k \sum_l |W_{k,l}|$

Elastic net (L1 + L2) $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$

Max norm regularization

Dropout

Batchnorm

Stochastic depth

If you're Bayesian: L2 regularization also corresponds to MAP inference using a Gaussian prior on W

Softmax Classifier (Multinomial Logistic Regression)

$$P(Y=k/X=x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}} \quad \text{where} \quad s = f(x_i; W)$$

$$L_i = -\log P(Y=y_i | X=x_i) \qquad L_j = -\log\left(\frac{e^{s_{yi}}}{\sum_j e^{s_j}}\right)$$

Want to maximize log likelihood, or to minimize the negative log likelihood of the correct class

Q: What is the min and max loss?

min = 0
max = infinity

Q2: Usually at initialization W is small so all $s \approx 0$ what is the loss?

$$= -\log\left(\frac{1}{C}\right)$$

Softmax vs SVM

Q: Suppose I take a datapoint and jiggle a bit (changing its score slightly). What happens to the loss in both cases?

SVM doesn't care about the score
Softmax continuously try to improve the datapoints, like pushing the correct to inf and pushing the incorrect to -inf

$$\text{Softmax}, L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

$$\text{SVM}, L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$\text{Full Loss } L = \frac{1}{N}\sum_{i=1}^{N} L_i + R(W)$$

Optimization:

Strategy #1 Random search

Strategy #2 follow the slope

In 1-dimension, the derivative of a function

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$ [slope]

In multiple dimensions, the gradient is the vector of partial derivatives along each direction is slope

- The slope in any direction is the dot product of the direction with the gradient
- The direction of steepest descent is the negative gradient.

→ This is super slow and super bad.

We can ~~use~~ compute anlytic gradient

Numberical gradient : approximate, slow, easy to write
Analytic gradient : exact, fast, error-prone

Gradient check: Using analytic gradient to find grads but checking with numerical gradient is
- This is an intersting debugging tool.

Gradient descent

i) find grads
ii) weights -= stepsize * grads
↑
hyperparamer
learning rate

Stochastic Gradient Descent:

Full sum is expensive when it's large
Aprrox sum using minibatch of examples
32/64/128 common

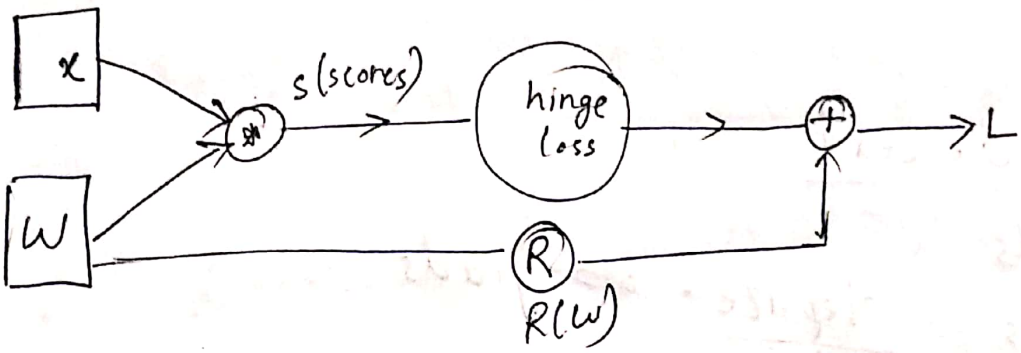
Image features:

Color Histogram
Histogram of Oriented Gradients
Bag of Words (Build codebook, Encode Images)

Image features vs ConvNets:

ConvNets (Krizhevsky 2012) AlexNet

Lecture 4: Intro to Neural Nets:



Leverage chain rule

$\frac{d}{dx}(e^x) = e^x$

$\frac{d}{dx}(ax) = a$

$\frac{d}{dx}\left(\frac{1}{x}\right) = -\frac{1}{x^2}$

~~$f(x) = c + x = a$~~

$\frac{d}{dx}(c+x) = 1$

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

$\frac{d\,\sigma(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2}$

$= \left(\frac{1+e^{-x}-1}{1+e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right)$

$= [1-\sigma(x)]\,\sigma(x)$

If any problem with gradients, break down to computational graph

add gate: gradient distributor

Q: What is a max gate?

the highest one back is gonna get the max value. other will be zeroed out

mul gate: gradient switcher

local gradient is the value of the other variable

Neural Nets:

2-layer Network $f = W_2 \max(0, W_1 x)$

Biological Neurons are far more complicated

Activations

Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$
Leaky ReLU	$\max(0.1x, x)$
tanh	$\tanh(x)$
Maxout	$\max(w_1^T x + b_1, w_2^T x + b_2)$
ReLU	$\max(0, x)$
ELU	$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$

$$\nabla_W f = 2q \cdot x^T$$

$$\nabla_W f = 2W^T q$$

Lecture 5: Convolution Neural Networks

- Try to maintain spatial structure

History of NN and CNN

→ Mark I perceptron machine was the first implementation of the perceptron algorithm.

* no backprop $f(x) = \begin{cases} 1 & w \cdot x + b \geq 0 \\ 0 & \text{otherwise} \end{cases}$

Frank Rosenblatt (1957)

→ Widrow and Hoff (1960): Adaline/Madaline

Multilayer perceptron ⇔ no backprop

Rumelhar (1986) → Backprop

[Hinton and Salakhutdinov 2006] →

Reinvigorated research in Deep Learning

First strong results [2012] [Krizhevsky, 2012]

- Imagenet classification with deep CNN

AlexNet

Neurocognition [Fukushima, 1980]

[Lecun, Botto, Bengio, Haffner, 1998] →
Gradient-based learning applied to document recognition

[K

ConvNets for classification, retrieval, detection, segmentation, self driving car, pose estimation, gameplay
Image Captioning
Neural Style Transfer

NVIDIA Tesla line [GPU] ~~powers~~

Convolutional Neural Networks:

Convolutional layer → preserve spatial structure

→ We call the layer convolutional because it is related to convolution of two signals

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

↑ elementwise multiplication and sum of a filter and the signal (image)

Conv Layer

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparams
  - $K \rightarrow$ # of filters
  - $F \rightarrow$ # their spatial extent
  - $S \rightarrow$ stride
  - $P \rightarrow$ Amound of zero padding

*Produces a volume of size $W_2 \times H_2 \times D_2$ where

- $W_2 = (W_1 - F + 2P)/S + 1$
- $H_2 = (H_1 - F + 2P)/S + 1$
- $D_2 = K$

→ with parameter sharing it introduces $F \cdot P \cdot D_1$ weights per filter for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases

* The output volume, the d-the depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S_2, and then offset by d-th bias.

Pooling layer

→ makes the representations smaller and more manageable

→ operates over each activation map independently

Pooling layer:

Accepts a volume of size $W_1 \times H_1 \times D_1$

- Requires three hyperparams
  - → their spatial extent f
  - → th strides
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - → $W_2 = (W_1 - f)/S + 1$
  - → $H_2 = (H_1 - f)/S + 1$
  - → $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for pooling layers

Fully Connected Layer (FC-Layer)

→ Contains neurons that connect to the entire input volume, as in ordinary Neural Networks

Typical architectures

[(Conv-ReLU)*N-POOL?]*M-(FC-ReLU)*K, Softmax

- ResNet/GoogleNet challenge this paradigm