Thursday, May 27, 2021 7:19 AM

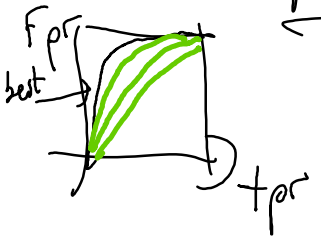
Neyman-Pearson Lemma

"Optimal" binary classifier is the likelihood ratio

best possible ROC curve

$$R(x) = \frac{P(x|S)}{P(x|B)}$$

$R(x) > R_c$ defines tpr & fpr



In lieu of a proof, let's check that $R(x)$ comes out of minimizing the binary cross entropy.

$$L = -\left(\sum_{i \in S} \log f(x_i) + \sum_{i \in B} \log(1 - f(x_i)) \right)$$

$f \to f + \delta f$

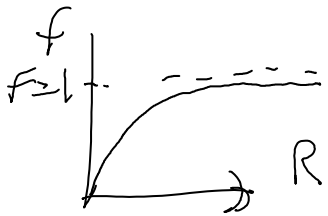
$$= -\int dx \left[P(x|S) \log f(x) + P(x|B) \log(1 - f(x)) \right]$$

$$0 = \frac{\delta L}{\delta f} = \frac{P(x|S)}{f(x)} - \frac{P(x|B)}{1 - f(x)}$$

Thursday, May 27, 2021 7:50 AM

$$F = \frac{P(x|S)}{P(x|S)+P(x|B)} = \frac{R}{R+1}$$

(monotonic wrt R)



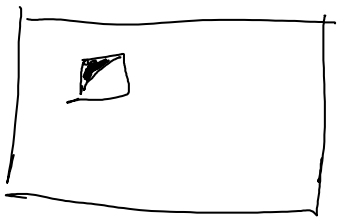
Convolutional Neural Network (CNN)

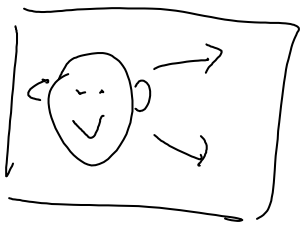
DNN is the most general feed forward NN, but has many weights

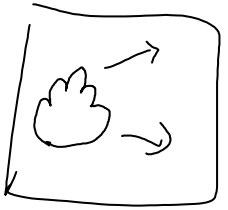
Can get better performance by reducing # of indep weights using symmetry.

For images, CNN architecture was a big breakthrough (2012)

- leverages 2d translation symmetry in images
- finds 2d "features" in images







Thursday, May 27, 2021 8:18 AM

To develop "feature detectors" or "feature maps" CNN will drag a linear filter over the image

↓

"convolution"

e.g. 2×2 filter  → trainable weights, will learn to detect specific feature in image (eg noses, faces, edges, ...)

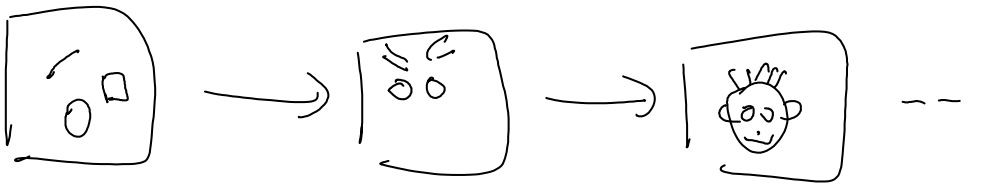
multiply pixels elementwise

$$\square = A\left(w_{11}a_{11} + w_{12}a_{12} + w_{21}a_{21} + w_{22}a_{22} + b\right) = a'_{11}$$

$$a'_{ij} = A\left(\sum_{r,s=1}^{2} a_{i+r,j+s}\, w_{rs} + b\right)$$

Thursday, May 27, 2021 8:24 AM

Can stack 2d convolutional layers like we do in DNN

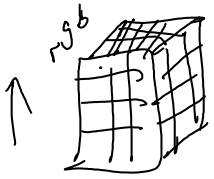


at every successive layer, CNN is learning higher level features in image

What about colors?

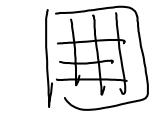
Can operate 2d convolution on multiple dimensions "channels"

(r, g, b)



multiply a filter map w/ 3 channels

sum over channels, output image w/ single channel.

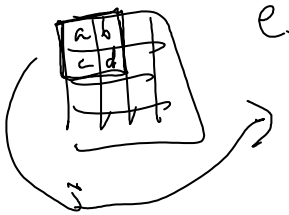


At each conv. layer, typically learn ~N ~100 filters

each conv layer then produces image w/ N channels.

Thursday, May 27, 2021 8:33 AM

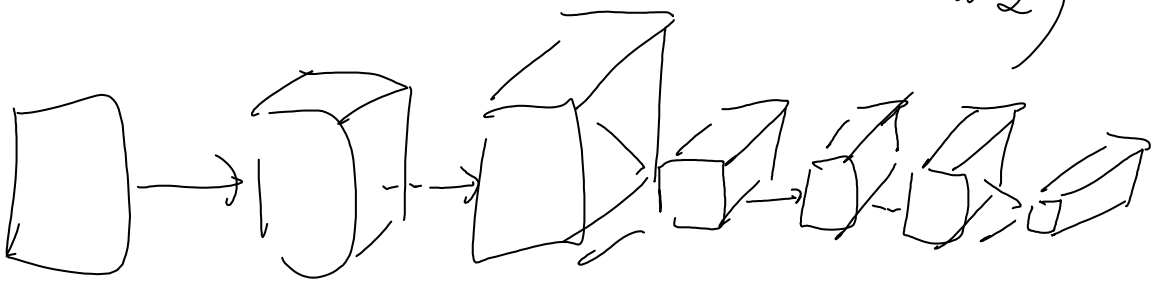
- Typically coarse grain image after some # of conv. layers
  "poling layer"



e.g. max pooling
Max (a, b, c, d)

avg pooling
$\frac{a+b+c+d}{4}$
etc.

→ reduce image size by some factor (e.g. factor of 2)



- Finally flatten reduced image to colum vector and feed to a DNN
  (reduced image: HLF's that CNN has learned; feed to a DNN: classifier for HLFs.)
- This is structure of the most basic CNN ("LeNet")

Thursday, May 27, 2021 9:06 AM

Generative Modeling & LHC applications

- Given some data $\{x_i\}$, can I learn $P(x)$ that data was drawn from and then sample from it to produce new examples of the data.
- 3 main approaches
  - Generative Adversarial Networks (Goodfellow et al 2014) ★
  - Variational Autoencoders
  - Normalizing Flows.

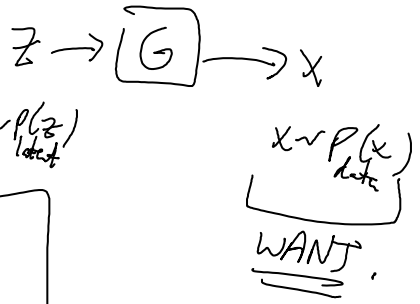
  } also relevant for next topic, anomaly detection

  → these give the best performance to date on image generation tasks

Thursday, May 27, 2021 9:16 AM

Idea: Adversarial training

- GAN has 2 NN's that are trying to defeat each other ("playing a game")
  1. G generator: generates fake data
  2. D discriminator: classifies real vs. fake.



- GAN loss: BCE btw generated & real data

$$L = \sum_{i \in \text{data}} \log D(x_i) + \sum_{i \in \text{gen}} \log\left(1 - D(G(z_i))\right)$$

- $z \in$ latent space w/ simple probability distn e.g. $P(z)$ = uniform or gaussian
  - $\mathbb{R}^d$ $d \ll$ dim of data
- L is negative of usual classifier loss, good clf. maximizes BCE. (usually)

Objective:

$$\min_G \max_D L$$

- $\max_D$: discriminator is as good as possible
- $\min_G$: generator fools discriminator as much as possible.

Thursday, May 27, 2021 9:24 AM

- Saddle pt optimization — tricky, unstable, how to guarantee convergence?

↓

you don't

GANs don't often converge

Select "best" epoch often based on subjective criteria.

↓

to train, typically alternate btw training G & D.

- train G for some # epochs fixing D
- vice versa

Thursday, May 27, 2021 9:28 AM

Optimality guarantee of GANs

$\min_G \max_D L$

$\max_D L|_G \longrightarrow \hat{D}_G(x)$ = optimal classifier btw real images & images generated by G.

$$= \frac{p_{data}(x)}{p_{data}(x) + p_G(x)}$$

"NP Lemma"

"Likelihood trick"

"Likelihood free learning"

plug back into L

$$L|_{D=\hat{D}_G} = \sum_{x \in data} \log \frac{p_{data}(x)}{p_{data}(x)+p_G(x)} + \sum_{x \in gen} \log \frac{p_G(x)}{p_{data}(x)+p_G(x)}$$

$$= \int dx\, p_{data}(x) \log p_{data}(x) + \int p_G(x) \log p_G(x) - \int \left(p_{data}(x)+p_G(x)\right) \log\left(p_{data}(x)+p_G(x)\right)$$

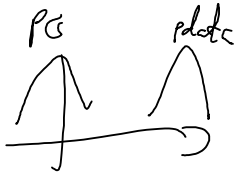
$$= 2\, JSD\left(p_{data}, p_G\right) + const.$$

"Jensen-Shannon Distance" measure of similarity btw 2 prob. distributions.

$JSD = 0$ iff $p_G = p_{data}$

$1 \geq JSD \geq 0$ for all p_G, p_{data}

$JSD = 1$ iff p_G & p_{data} are disjoint →



$\min_G L|_{\theta=\hat{\theta}_G} \Rightarrow JSD = 0$ i.e. $p_G = p_{data}$