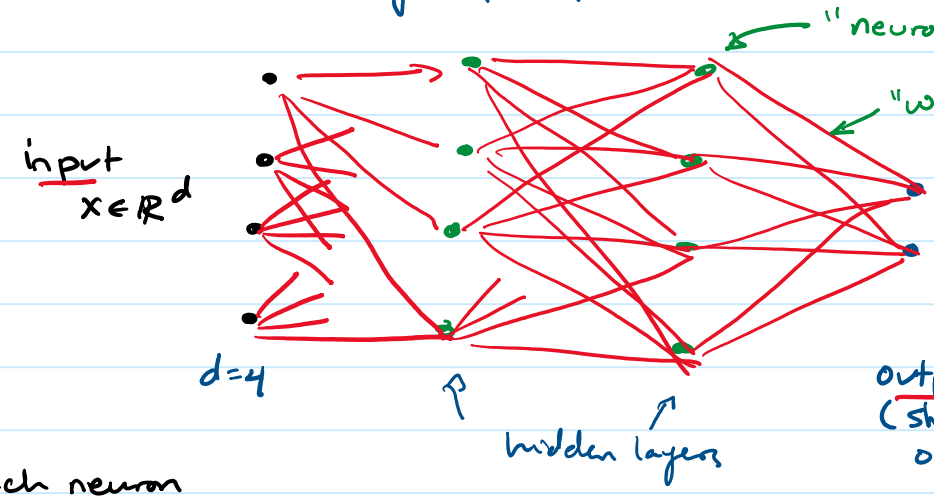
(Artificial) Neural Networks

Sunday, August 25, 2024 9:56 AM

Neural Nets simplest kind is "fully-connected feed-forward"

aka "multi-layer perceptron" MLP



each neuron applies a nonlinear activation function σ to a linear combination of its inputs

mathematically, (hyperparameter)

$$x^{(1)} = \sigma\left(W^{(1)} \cdot x + b^{(1)}\right)$$
$$x^{(2)} = \sigma\left(W^{(2)} \cdot x^{(1)} + b^{(2)}\right)$$
$$y = \sigma\left(W^{(3)} x^{(2)} + b^{(3)}\right)$$

(σ applied componentwise)

Training means choosing weights $W^{(k)}$ and biases $b^{(k)}$

Activation functions: ReLU (rectified linear unit), $\sigma(a) = \max(a, 0)$ 

sigmoid, "GeLu", ... also common choices

Approximation Error

- Weierstrass

https://en.wikipedia.org/wiki/Stone%E2%80%93Weierstrass_theorem

The set of polynomials is dense in $C([a,b])$ w.r.t. $\|\cdot\|_\infty$

i.e. $\forall f \in C([a,b])$, $\forall \varepsilon > 0$, $\exists$ polynomial p s.t. $|p(x) - f(x)| < \varepsilon \quad \forall x \in [a,b]$

Universal Approximation Thm

A spate of papers in the 1980s—1990s, from George Cybenko and Kurt Hornik [de] etc, established several universal approximation theorems for arbitrary width and bounded depth.[37][3][38][4] See[39][40][6] for reviews. The following is the most often quoted:

> **Universal approximation theorem** — Let $C(X, \mathbb{R}^m)$ denote the set of continuous functions from a subset X of a Euclidean $\mathbb{R}^n$ space to a Euclidean space $\mathbb{R}^m$. Let $\sigma \in C(\mathbb{R}, \mathbb{R})$. Note that $(\sigma \circ x)_i = \sigma(x_i)$, so $\sigma \circ x$ denotes σ applied to each component of x.
>
> Then σ is not polynomial if and only if for every $n \in \mathbb{N}$, $m \in \mathbb{N}$, compact $K \subseteq \mathbb{R}^n$, $f \in C(K, \mathbb{R}^m)$, $\varepsilon > 0$ there exist $k \in \mathbb{N}$, $A \in \mathbb{R}^{k \times n}$, $b \in \mathbb{R}^k$, $C \in \mathbb{R}^{m \times k}$ such that
>
> $$\sup_{x \in K} \|f(x) - g(x)\| < \varepsilon$$
>
> where $g(x) = C \cdot (\sigma \circ (A \cdot x + b))$

Fancy Neural Networks

Sunday, August 25, 2024 4:41 PM

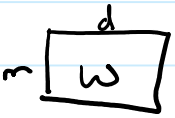
Convolutional Layers

Basic idea: require the weights W to be a convolution (in 1D, 2D, ...)

Why?
- Fewer parameters — Can prevent overfitting; Can speed up code
- Often makes sense due to prior knowledge
  - very common in image applications
  - Plain convolution is translation equivariant
    - i.e. translate input → translate output (Equivariance)
    - (translate input → no effect (Invariance))

Input: $x \in \mathbb{R}^d$

Output: $y \in \mathbb{R}^m$ Fully connected: $y = W \cdot x$



$m \cdot d$ parameters

Simplest convolution: w/ filter (aka kernel) $h \in \mathbb{R}^d$,

Output is size $m \approx d$ ($m = d$ if periodic)

$$y = h * x \qquad y_i = \sum_j x_{i-j} \cdot h_j = \sum_j x_j h_{i-j}$$

(I'm ignoring "boundary conditions")

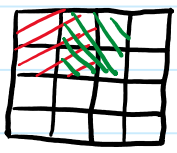
So convolution has d parameters instead of d^2 (for fully connected)

Typically h only has a few nonzeros (kernel size), say k, so in fact convolution has then just k parameters and takes just $O(k)$ flops

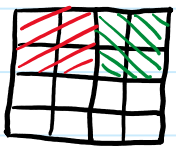
Practical Considerations

- Usually have several filters "in parallel", creating new "channels"
- Parameters per kernel:
  - padding / boundary conditions
  - stride
  - filter size

2D conv, size 2x2



... Stride = 1



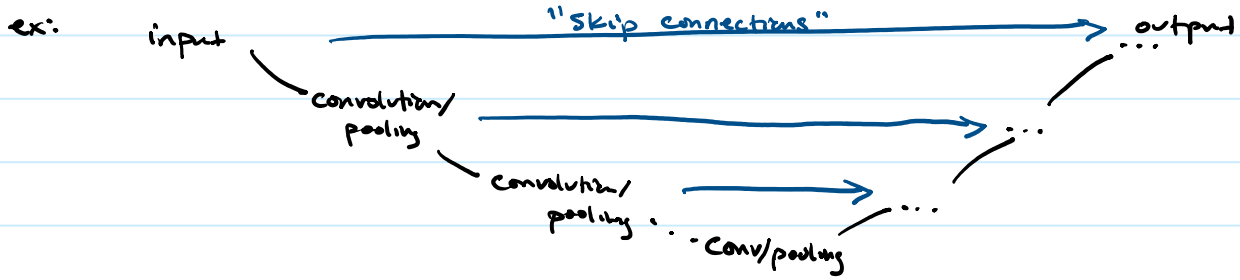
... Stride = 2

- Often followed by a pooling operation (eg. max-pooling) to downsample (must choose size)

Fancy Neural Networks: U-nets, Resnets, AutoEncoders

Tuesday, August 27, 2024 4:26 PM

U-nets A type of CNN often used in medical imaging



Resnets (Residual Networks) / skip connections

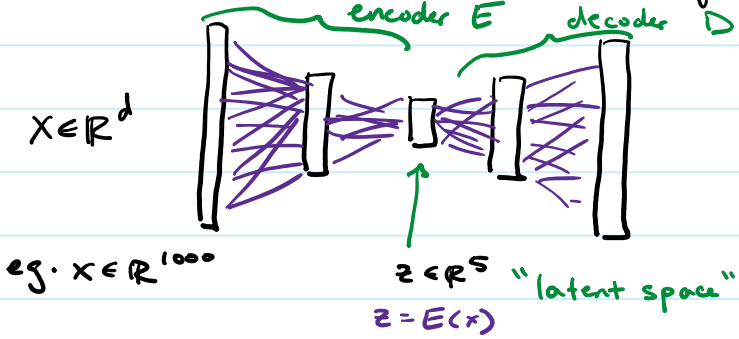
Instead of $X^{(k+1)} = \sigma(W \cdot X^{(k)} + b)$

do $X^{(k+1)} = X^{(k)} + \sigma(W \cdot X^{(k)} + b)$ ← identity; ← perturbation to identity

Sometimes do $X^{(k-3)}$ or something

Helps "stabilize" training if used appropriately, can help w/ vanishing gradients

Autoencoders (AE) have a **bottleneck** layer



AE are self-supervised:

loss is $\| D(E(x)) - x \|^2$

ideally $D \circ E$ is identity

(but it can't be that for all inputs due to bottleneck, i.e. $5 < 100$)

We'll later talk about

Variational AE (vAE) a type of generative model

(as well as **GANS** and **diffusion models**)

Fancy Neural Networks: Recurrent and Transformers

Tuesday, August 27, 2024 4:46 PM

Recurrent Neural Nets (ex: LSTM)

Helps with two (related) issues: 1) what if not all X are the same size?

2) how to encode positional structure?

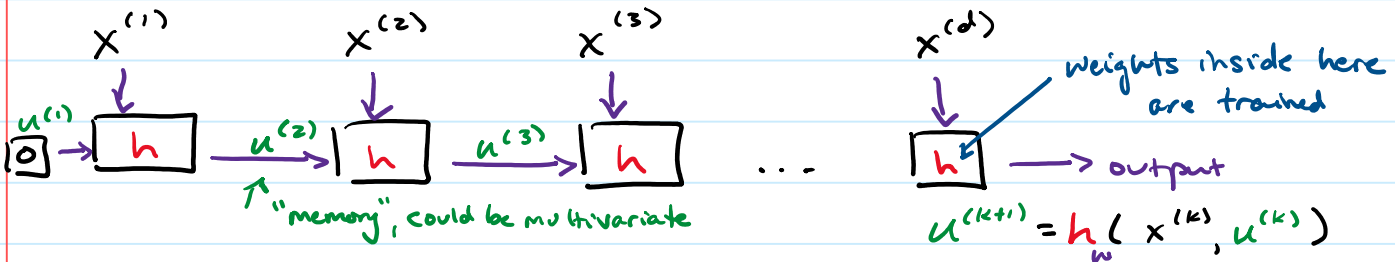
ie. standard ML: $X_i = \begin{bmatrix} \#\text{typos in email} \\ \#\text{all capital words in email} \end{bmatrix} \in \mathbb{R}^2$ ← position doesn't matter

vs. time-series ML: $X_i =$ (the entire email) ← position matters

Ex: NLP, time-series (robotics, guidance-navigation-control)

RNN deal with this using special layers.

Let $X \in \mathbb{R}^d$ ← "d" depends on X (ex: $X = [\text{Hello\_world}]$, with $x^{(1)}$, $x^{(2)}$, $x^{(11)}$)



Transformers and Attention mechanism

"Attention is all you need" 2017 Vaswani et al. 10^4+ citations

Also popular in NLP

Encodings
- old-school: "bag of words", "one hot"
- large: digrams, trigrams {"He", "el", "ll", "lo", "o_", ...} large!
- RNN style
- positional encodings: word encoded as a vector (word embedding) and location in sentence encoded as vector

	Index of Token	Positional encoding			
Hello	0	P_{00}	P_{01}	...	P_{0d}
World	1	P_{10}	P_{11}	...	P_{1d}
today.	2	P_{20}	P_{21}	...	P_{2d}

d is parameter you choose

$$P_{k,2i} = \sin\left(\frac{k}{n^{2i/d}}\right), \quad P_{k,2i+1} = \cos\left(\frac{k}{n^{2i/d}}\right)$$

$n = 10^4$ a parameter you set