TAM 598 Lecture 24:

Neural Networks

Announcements:

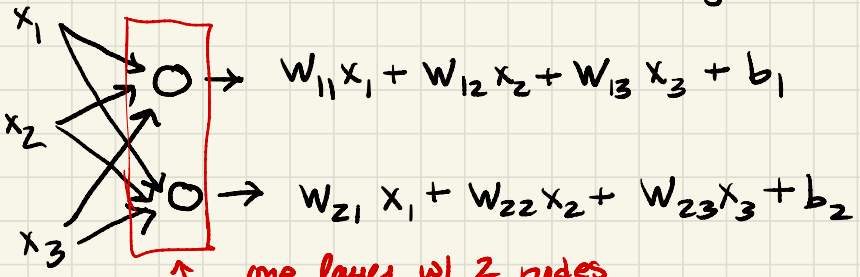
- HW 6 covers lectures 21-23; due on Fri May 2

Neural Networks - massively parametrized function approximators that express hierarchically layered information (each successive layer captures increasingly abstract features from the input)

→ operate in layers

→ each layer takes as input the output of the previous layer, and

- ⊙ applies a linear transformation
- ⊙ adds a constant bias
- ⊙ passes result through a non-linear function
- ⊙ passes to the next layer

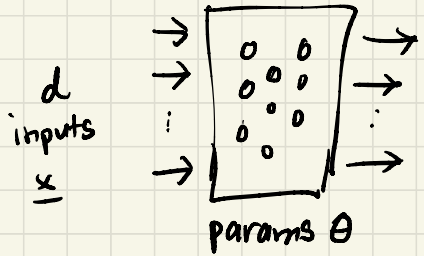


$\underline{\underline{W}}\,\underline{x} + \underline{b}$

$\underline{\underline{W}}$ is (3×2)

$\underline{b}$ is (1×2)

"deep" ⇒ many layer (≥ 5 ish)



$$\underset{\sim}{y} = f(\underset{\sim}{x}; \theta)$$

parameters

one layer: $\underline{y}_1 = h^{(1)}(\underline{\underline{W}}^{(1)} \underline{x} + \underline{b}^{(1)})$ $\qquad \underline{y}_1 = f_1(\underline{x})$

two layers: $\underline{y}_2 = h^{(2)}(\underline{\underline{W}}^{(2)} \underline{y}_1 + \underline{b}^{(2)})$ $\qquad y_2 = f_2(\underline{y}_1) = f_2(f_1(\underline{x})) = f_2 \circ f_1(\underline{x})$

$\vdots$

L layers $\qquad \underline{y}_n = (f_L \circ f_{L-1} \cdots \circ f_1)(\underset{\sim}{x})$

"fully connected DNN"

$h^{(i)}$ is an activation
a nonlinear function applied to each argument
(ReLU, sigmoid, hyperbolic tangent, sinusoid, step function)

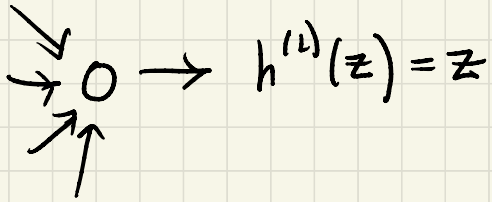
$\underline{\underline{W}}^{(i)}$ is a weight matrix

$\underline{b}^{(i)}$ is a bias

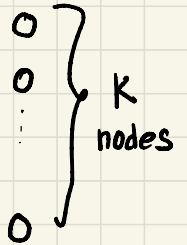
other archs:
recurrent
CNNs
autoenc
residual NNs

Most typically all but the last layer have the same activation function. The last depends on constraints of the final output, eg

1) a real number



2) pmf on K categories



$$h^{(L)}(\underline{z}) = \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)}$$

universal approximation theorem - given a good activation function, and enough nodes, we can approximate ANY continuous function arbitrarily well

---

Training Regression Networks - Loss Function View

inputs $\underline{x}_{1:n} = (\underline{x}_1, \ldots, \underline{x}_n)$

outputs $\underline{y}_{1:n} = (y_1, \ldots, y_n)$

$$y = f(\underline{x}; \theta)$$

→ need to fit these params (weights, biases)

Loss Function $$L(\theta) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - f(\underline{x}_i; \theta) \right)^2$$

Find $$\theta^* = \operatorname{argmin} L(\theta)$$

solution: not unique, not linear, not convex, ...

local min is a global min, lots of algs guaranteed to converge

Training Regression Networks – Probabilistic View

likelihood $p(y_{1:n} \mid \underline{x}_{1:n}, \theta)$

maximize log likelihood $\mathcal{L}(\theta) = -\log p(y_{1:n} \mid \underline{x}_{1:n}, \theta)$

same result as before if observations are independent and measurement noise is gaussian w/ mean given by the NN and constant variance

ie say $p(y_i \mid \underline{x}_i, \theta) = \mathcal{N}(y_i \mid f(\underline{x}_i, \theta), \sigma)$

$$= \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{ - \frac{(y_i - f(\underline{x}_i; \theta))^2}{2\sigma^2} \right\}$$

from independence $p(y_{1:n} \mid \underline{x}_{1:n}, \theta) = \prod_{i=1}^{n} p(y_i \mid x_i, \theta)$

So we want to minimize

$$L(\theta) = -\log p(y_{1:n} | x_{1:n}, \theta)$$

$$= -\sum_{i=1}^{n} \log p(y_i | x_i, \theta)$$ same!

$$= \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - f(x_i, \theta))^2 + \text{const}$$

Stochastic Optimization ⇒ subsample available data (in batches)

issues: $L(\theta)$ is non convex
summation in $L(\theta)$ could be huge

$$\min_{\theta} \mathbb{E}_z [\ell(\theta; z)]$$

minim. exp. over z of $\ell(\theta, z)$

z = random vector

going to be idx of data pt

eg let $I \sim \text{Categorical}\left(\frac{1}{n}, \dots, \frac{1}{n}\right)$

pick w/ equal prob the index of one of the n obs.

then $\ell(\theta, I) = \left(y_I - f(\underline{x}_I; \theta)\right)^2$ so $Z = I$

and $E_I\left[\ell(\theta, I)\right] = \sum_{i=1}^{n} p(I=i)\, \ell(\theta, i)$

$$= \sum_{i=1}^{n} \left(y_i - f(\underline{x}_i; \theta)\right)^2$$

$$= L(\theta)$$

minimizing $E_I[\ell(\theta, I)]$ is same as minimizing $L(\theta)$

eg w/ batch of size m

let $I_1, I_2, \dots, I_m$ all be Cat $\left(\frac{1}{n}, \dots, \frac{1}{n}\right)$

then $l_m\left(\theta; I_{1:m}\right) = \frac{1}{m} \sum_{j=1}^{m} \left(y_{I_j} - f\left(\underline{x}_{I_j}; \theta\right)\right)^2$

and $\mathbb{E}\left[l_m(\theta; I_m)\right] = \mathbb{E}\left[\frac{1}{m} \sum_{j=1}^{m} \left(y_{I_j} - f(\underline{x}_{I_j}; \theta)\right)^2\right]$

$= \frac{1}{m} \sum_{j=1}^{m} \left[\mathbb{E}\left(y_{I_j} - f(\underline{x}_{I_j}; \theta)\right)^2\right]$

$= \frac{1}{m} \sum_{j=1}^{m} L(\theta)$

$= L(\theta)$

Stochastic Gradient Descent (aka Robbins-Monro algorithm)

want $\min_\theta \mathbb{E}_z[\ell(\theta; z)]$

algorithm: ⊙ initialize $\theta = \theta_0$

⊙ iterate $\theta_{t+1} = \theta_t - \alpha_t \underbrace{\nabla_\theta \ell(\theta_t, z_t)}_{\text{noisy gradient}}$ where

z_t are independent samples of z

α_t is the **learning rate**

as long as $\sum_{t=1}^{\infty} \alpha_t = +\infty$

$\sum_{t=1}^{\infty} \alpha_t^2 < +\infty$

} then SGD converges to a local min of $\mathbb{E}[\ell(\theta, z)]$

eg $\alpha_t = \frac{A}{(Bt+C)^e}$ with $e \in (0.5, 1)$

so with batches of size m at step t:

choose m indices randomly $i_{t1}, i_{t2}, \dots, i_{tm}$

$$\theta_{t+1} = \theta_t - \alpha_t \nabla_\theta \frac{1}{m} \sum_{j=1}^{m} \left(y_{i_{tj}} - f\left(\underline{x}_{i_{tj}}; \theta\right)\right)^2$$

$$= \theta_t - 2\alpha_t \frac{1}{m} \sum_{j=1}^{m} \left(y_{i_{tj}} - f\left(\underline{x}_{i_{tj}}; \theta\right)\right) \nabla_\theta f\left(\underline{x}_{i_{tj}}; \theta\right)$$

we get the gradient $\nabla_\theta f$ using back-prop / autodiff

other stochastic opt alg; SGD w/ momentum
AdaGrad
Adam (adaptive moment estimation)