# TAM 598 Lecture 24:

## Neural Networks

Announcements:

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

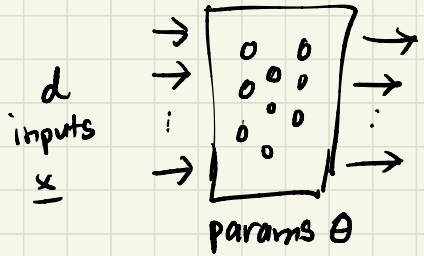
Neural Networks - massively parametrized function approximators that express hierarchically layered information (each successive layer captures increasingly abstract features from the input)

→ 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

"deep" ⇒ many layer ( ≥ 5 ish)




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

parameters

one layer:

two layers:

L layers

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

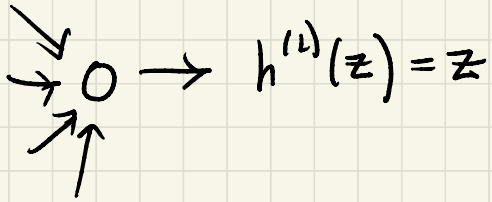
$\underline{\underline{W}}^{(i)}$ is a weight matrix

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

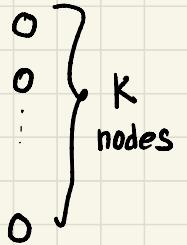
---

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




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

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

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

# 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

So we want to minimize

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

$$= -\sum_{i=1}^{n} \log p(y_i | \underline{x}_i, \theta)$$

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

---

Stochastic Optimization

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

then $\ell(\theta, I) =$

and $E_I\left[\ell(\theta, I)\right] =$

eg w/ batch of size $m$

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

then $\ell_m\left(\theta; I_{1:m}\right) =$

and $\mathbb{E}\left[\ell_m(\theta; I_m)\right] =$

# Stochastic Gradient Descent (aka Robbins-Monro algorithm)

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

algorithm: ⊙ initialize $\theta = \theta_0$

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

$z_t$ are independent samples of $z$

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)