# Lec 17

Tuesday, November 12, 2019 10:59

## Recap: Neural Nets

Up until now, for linear models

$$\hat{f}(x) = g(\beta^T \varphi(x))$$

$\varphi$ given, $\beta$ learned

Big idea of NN: also learn $\varphi$

Vanila NN:

$$\underset{\mathbb{R}^P}{X} \longrightarrow \underset{\mathbb{R}^M}{Z = \varphi(X)} \longrightarrow \hat{f}(X) = \beta^T Z$$

$$\varphi_m(X) = \sigma(\alpha_{0m} + \alpha_m^T X) \quad m = 1, \dots, M$$

Given $\alpha$'s, $\beta$'s, the steps to compute the output of the vanila neural network:

- Get input $x \in \mathbb{R}^P$
- Compute hidden layer $z_j = \sigma(\alpha_{j0} + \alpha_j^T x)$
- Compute outputs $T_k = \beta_{k0} + \beta_k^T z$
- For regression, just return outputs
- For classification, apply softmax & return probabilities

$$\hat{P}_k = \frac{e^{T_k}}{\sum_{k'} e^{T_{k'}}}$$

The params of this model are $\alpha$'s, $\beta$'s

Want to find params that min loss on training data / max likelihood

For regression: min sum of squared errors

$$\min_{\alpha,\beta} \sum_i \sum_k (Y_{ik} - \underbrace{T_{ik}}_{\text{dependence on } x_i,\, \alpha,\beta})^2$$

For classification: max likelihood / min cross entropy

$$\min_{\alpha,\beta} -\sum_i \sum_k Y_{ik} \log \underbrace{\hat{p}_{ik}}_{\text{depends on } \alpha,\beta,x_i}$$

Can add ridge regularization:

$$\min_{\text{params}} \text{loss} + \lambda \underbrace{\|\text{params}\|_2^2}_{\alpha_{11}^2 + \alpha_{12}^2 + \dots + \alpha_{1p}^2 + \alpha_{21}^2 + \dots + \beta_{11}^2 + \dots}$$

called "weight decay"

How to actually solve $\min_{\text{params}}$ loss ?

Put NN to the side.

Let's tackle general problem

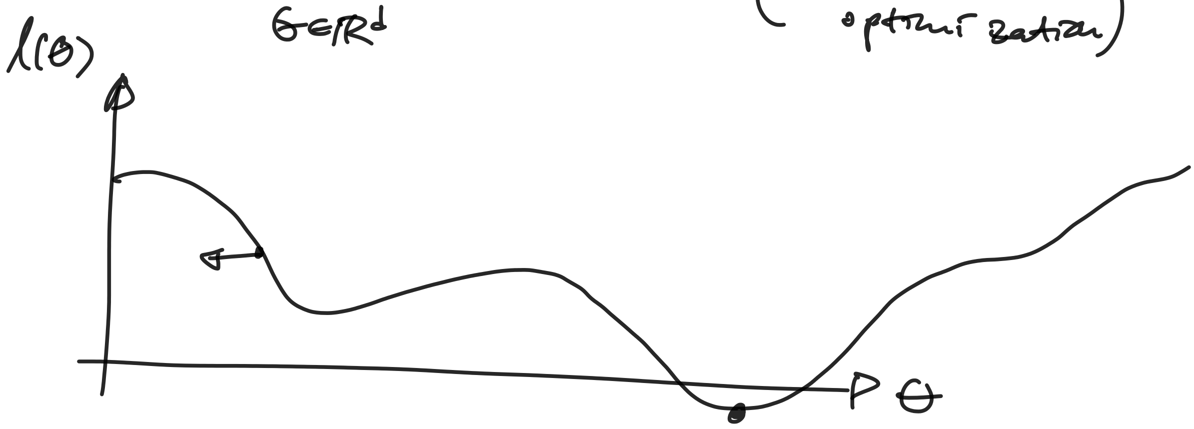
## Gradient Descent & Variants

Given a function $l(\theta)$, $\theta \in \mathbb{R}^d$

$$l: \mathbb{R}^d \to \mathbb{R}$$

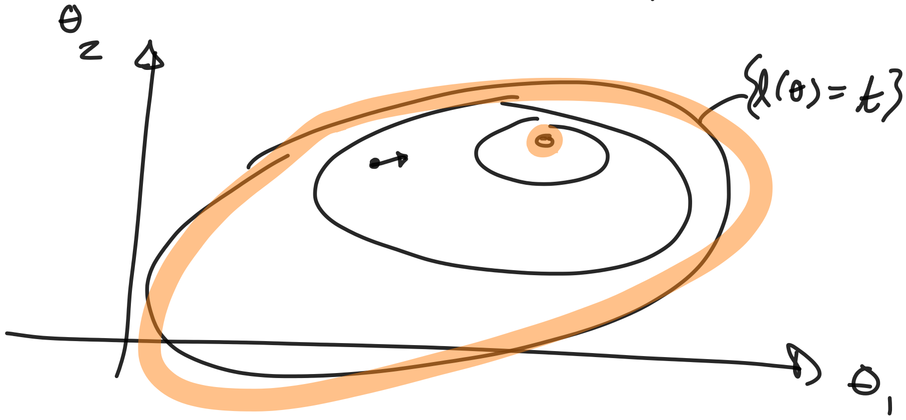
Want to solve

$$\min_{\theta \in \mathbb{R}^d} \ell(\theta) \quad \text{(unconstrained optimization)}$$




Recall: $\nabla \ell(\theta) = \begin{pmatrix} \partial \ell(\theta)/\partial \theta_1 \\ \vdots \\ \partial \ell(\theta)/\partial \theta_d \end{pmatrix} \in \mathbb{R}^d$

$$\nabla \ell : \mathbb{R}^d \to \mathbb{R}^d$$




Gradient = direction of highest ascent

To min, want to go down

− Grad = direction of greatest descent

Idea: to min a fn (i.e. find its lowest pt in the graph), walk in the direction of greatest descent

## Vanila Gradient Descent Algo

Inputs: − Starting pt $\theta_0 \in \mathbb{R}^d$

− Gradient fn $\nabla \ell(\theta)$

- Step size $\alpha$
- Tolerance $\tau$, $m$

For $t = 1, 2, \cdots$

$$\theta_t \leftarrow \theta_{t-1} - \alpha \nabla l(\theta_{t-1})$$

If $\|\theta_{t-i} - \theta_{t-i-1}\| < \tau \quad \forall i = 0, \cdots, m-1$

Return $\theta_t$

Variations on choosing stepsize $\alpha$

One variation: Line search

$$\alpha_t = \underset{\alpha_t \geq 0}{\operatorname{argmin}}\, l(\theta_{t-1} - \alpha \nabla l(\theta_{t-1}))$$

$$\theta_t = \theta_{t-1} - \alpha_t \nabla l(\theta_{t-1})$$

Now suppose $l(\theta) = \frac{1}{n} \sum_{i=1}^n l_i(\theta)$

e.g. $l_i(\theta) = (y_i - \hat{f}(x_i; \theta))^2$

$$\nabla l(\theta) = \frac{1}{n} \nabla \sum_i l_i(\theta)$$

$$= \frac{1}{n} \sum_i \nabla l_i(\theta)$$

In words: "the gradient of the average is the average of the gradients"

To run gradient descent we'd need to compute $\nabla l_i(\theta)$ for $i = 1, \cdots, n$

In order to get $\nabla L(\theta)$

Just for one step

Maybe too much when $n$ is very large

## Stochastic Gradient Descent

1. Draw $i \sim \text{unif}\{1, \cdots, n\}$
2. $\theta_t \leftarrow \theta_{t-1} - \alpha \nabla l_i(\theta_{t-1})$

Other var.