11-785 01/05/2020

Lecture 3 - Training NN

- main points; instructor intuition; areas to review.

Recap

- (*) minimal architecture constraints on NN capabilities
- I/O representation → later in course
- (Q) How do we compose the network that performs requisite function?

- Perceptron as basic unit, fires/turns output to 1 (threshold)
- modify from weighted combo → weighted combo + bias (affine)

BR: what is formal definition for sth to be linear (linearity in parameters): $f(ax+by) = af(x)+bf(y)$ (w)

- Affine $f(x)$: $\exists C, g(x) = f(x) - C$
  $g(x)$ - linear
- (*) Affine combination of inputs + activation

$y = \sigma(z) \quad z = \sum_i w_i x_i$

- Remember modification of bias

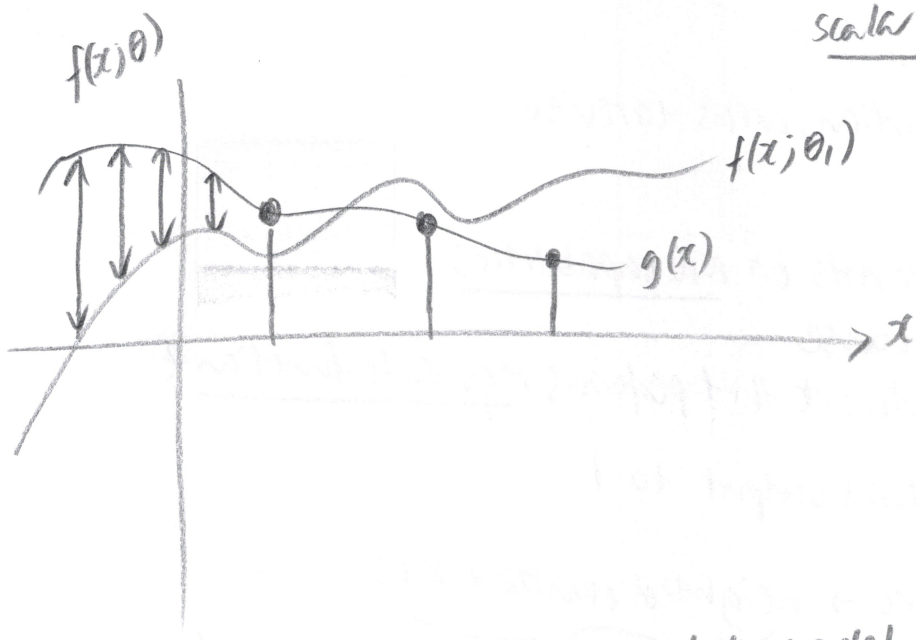
- Feedforward architecture: No feedback into inputs.
  - Assume network architecture is sufficient

- Network is a determ. function
- Parameters of network - weights, biases
- Parameters must be set (weights, biases)

(Q): How to construct MLP to represent a function?

1. By hand (A1) → check the by-hand intuition
2. Automatic tuning of parameters of network

scalar illustration



- Diagram not fully accurate.

- BR: Quantify gap between function we want to model $g(x)$ and our NN estimated fn; and area between curves.

- $f(X;\underline{W})$ has capacity to rep. $g(X)$

$$\hat{\underline{W}} = \underset{\underline{W}}{\operatorname{argmin}} \int_X div(f(X;\underline{W}), g(X))\, dX$$

- This assumes $g(X)$ is known at every X.
- (*) In practice; $g(X)$ is not specified, only samples from $g(X)$
- Only noisy observations/samples from $g(X)$ i.e. input-output pairs

---

(*) estimate/train NN: Determine weights/biases required for it to model a desired function

---

Original MLP (binary classification) (History)

- fairly comfortable with most of this so far; many not need review
- Restating perceptron

- Pad data with 1

$$y = \begin{cases} 1 \text{ if } \sum_{i=1}^{N+1} w_i x_i \geq 0 & x_{N+1} = 1 \\ 0 \text{ otherwise} \end{cases}$$

- Boundary → hyperplane through origin
- Can't remember perceptron exactly, but am clear on what happens

- A lot of exposition on perceptron

Novikov perceptron theorem (bound on convergence) - $\left(\frac{R}{\gamma}\right)^2$ misclassi

- Bound on classification

Illustration of how perceptron algorithm to model complex decision boundary (Heuristic) (not comput. feasible)

- Exponential input search, relabelling

(*) Training MLP using perceptron → combinatorial opt.
NP, exponential complexity., intermediate labels
(*) Do not know outputs of indiv. intermediate neurons in net for train. input

---

- Historically → ADALINE/MADALINE (greedy algorithm)
- (W)(A1) - Post lec review - done
- (W)(A2): - clarity on exponential time complexity/perceptron failure for complex decision boundary (*)

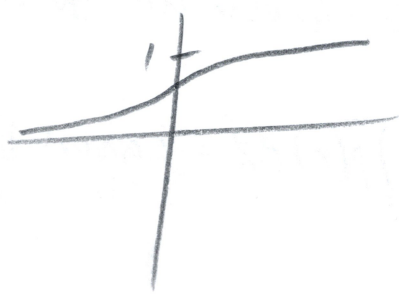
(*) BR: MLP as combinatorial opt → stifled research
(+) Issue lies in threshold function
(*) No way of knowing whether small parmeter perturbations reduce error (i.e. an issue to do with information gained from feedback)
- i.e. individual neurons weights can change significantly without changing error.
- to simple MLP: derivatives of function with respect to weights 0 nearly everywhere; no derivatives at boundaries

(W)(A3) - Review for clarity

Issue 2: Real-life data 'not linearly separable' (~)

Sol: make activation function differentiable
(*) Replace threshold activ in perceptron with sigmoid activations:-

$$\sigma(z) = \frac{1}{1+e^{-z}}$$



- can make $\sigma(z)$ approximate threshold

- BR: sigmoid activation
  i) tells me if I'm classifying correctly
  ii) gives an indication of how close I am.

- Non-linearly sepcable data
  . even in 10 cases - not possible to threshold counterexamples

---

- sigmoid probabilistic interpretation $p(y=1|\underline{x})$ (recall classification)
- logistic regression: (W)(14) - refresh
  ↳ Perceptron with sigmoid activation (pr input belongs to class 1)

---

(22) - key slide

- $\sigma(z)$ - differentiable fn. $\frac{dy}{dz} = \sigma'(z)$
- Diff output y wrt:-
  i) inputs x_i $\frac{dy}{dw_i} = \frac{dy}{dz}\frac{dz}{dw_i} = \sigma(z)x_i$
  ii) param w_i $\frac{dy}{dx_i} = \frac{dy}{dz}\frac{dz}{dw_i} = \sigma'(z)w_i$
- via chain rule y is a diff function of both inputs x_i and weights x_i
(*) can compute change in output for small changes in either inputs or weights (W): Think turning knobs physically

---

- same holds for more complex networks.
(W)(15) - Review check you understand the examples (quick)

- If activations differentiable, overall function is different wrt every param.

BR:

- sets stage for training MLP from input output examples.
  $(\underline{X}_1, d_1)$; $(\underline{X}_2, d_2)$ ... ; $(\underline{X}_N, d_N)$
- From earlier ; modify (W)(16) :- Go over wrt ~~to~~ some eleven. Bishop
- Assume $\underline{X}$ is a r.v.

$$\hat{W} = \arg\min_W \int_X div(f(X;W), g(X))\, p(X)\, dX = \arg\min_W \mathbb{E}_{X\sim p(X)}[div(f(X;W), g(X)]$$

- BR: His presentation misses out overfitting, expected error x

---

- Expected error:

$$\mathbb{E}[div(f(x;w),g(x))] = \int_x div(f(x;w),g(x))\,p(x)\,dx$$

BR: Average error over entire input space

- Empirical est:

$$\mathbb{E}[div(f(x;w),g(x))] \approx \frac{1}{N}\sum_{i=1}^{N} div(f(x_i;w),d_i)$$

BR: Average error over samples

- (w)(17) - fit this in with your existing understanding (in Bishop)

(*): minimise empirical risk i.e. go for ERM

Given $(x_1,d_1)\ldots,(x_N,d_N)$

- error on i^{th} distance:- $div(f(x_i;w),d_i)$
- empirical average error (empirical risk) on all training data:-

$$loss(w) = \frac{1}{N}\sum_i div(f(x_i;w),d_i)$$

- Estimate param. to min empirical est of exp. risk:-

$$\hat{w} = \arg\min_w loss(w)$$

- min empirical risk over training samples

(*) Instance of function minimisation → optimisation (non-convex)

(*) differentiability → Δparam effect on output → effect on error

---

optimisation crash course

- usual account of a derivative through limits, infinitesimals
- some issues with Leibniz notation (he brings up)
- Presentation of gradient operator (engineering approach) - I'm okay thanks

$$\nabla_x y = \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \vdots \\ \frac{\partial y}{\partial x_D} \end{bmatrix}$$

BR

- skips (w)(18) - review gradients

(41)(49) - are you comfortable with rate of increase of function (scalar) and gradient

(*) - Inner product justif. of gradient

(*) Gradient of a scalar function of a vector points in direction of rate of greatest increase of function; -ve rate of fastest decrease of function