Convolutiona Networks Basic (Overview)

MIchaelle Perez

April 25, 2021

Contents

1 Learning Algorithms

1.1 Traditional AI vs ML

- Traditional AI tries to build an algorithm in the strict sense of the word that solves a problem.
- ML try to learn from Data i.e. examples.
  - ML is based on creating a optimization problem in a suitable space.
  - We only need to crate a measure of success and search algorithm on the solution space.

1.2 Why ML

- ML can solve tasks that are too difficult to solve with a fixed program (classical AI).
- There is a profound link between **understanding** ML and understanding the principles of human learning (intelligence).

1.3 Tasks – Examples

1.3.1 Examples

- A ML model is described by how it handles examples.
- An example is an element on feature space.
  - Features are any kind of mathematical object.

- The feature space defines the domain of the underlying function that describes the model.
- Features can be very well hand crafted (specific measures, characteristics, etc.) or not so well crafted (picture).

1.3.2 Common Tasks

- Regression: $f : \mathbb{R}^n \to \mathbb{R}$, samples $f(x_i) = t_i$.
- Classification: $f : \mathbb{R}^n \to \{1, \dots N\}$ samples $f(\mathbf{x}_i) = t_i$.
  - Object recognition [ref]
  - Face recognition [ref]
- **Classification with Missing Inputs:**

$$\{f_i : \mathbb{R}^{n_i} \to \{1, \dots, N\},$$

  samples $f_i(\mathbf{x_j}) = t_j$.
  - Zero shoot learning [ref]
  - Super resolution [ref]

1.3.3 Common Tasks

- Denoising: $f : \mathbb{R}^n \to \mathbb{R}^n$, samples $f(\hat{\mathbf{x}}) = \mathbf{x}$.
  - Density estimation.
  - Clustering.
- DE: $f : \mathbb{R}^n \to \mathbb{R}^n \times \mathbb{R}^m$ samples $f(\mathbf{x}) = [\mathbf{x} \overset{u}{\mapsto} u(\mathbf{x})]$.
  - ??
  - ??

1.4 How to do it?

1.4.1 Measuring Performance

- A key point in the design of a ML algorithm for a specific task T is find a **quantitative measure of the performance**.
- As an example in the classification task:

- We can measure accuracy over a group of samples:

$$\text{Acc} = \frac{1}{N} \left|\{\mathbf{x}_j \mid f(\mathbf{x}_j) = t_j\}\right|$$

- The real measure of success of a ML model is **how well it behaves with data that it has not been seen before**.

1.4.2 Supervised and Unsupervised Learning

2 Main Model Deep Feed Forward Networks

2.1 Hypothesis

- Work on AI problems (specially classification) as a parametric statics.
- Given the existence of a $f(X)$ s.t. $f(X_i) = t_i$ for a given set $D = \{(X_i, t_i)$.
- Given a family of functions $f(X, \theta)$ where X is the input and θ is the parameter. The goal is to find θ^* s.t. $f(X, \theta^*)$ is close to f.
- We must study how to find the θ_i the nature of the $f(\cdot, \cdot)$ and $D = \{(X_i, t_i)\}$.

2.2 The "family" of $f(\cdot, \cdot)$

- We cant make a search over all the possible functions $f(\cdot, \cdot)$, we *must* restrict our search to a certain kind of functions (**inductive bias**).
- The feed forward networks work using a simple approach:

$$f(X_i, \theta) = f_1(f_2(f_3(\dots(f_n(X_n, \theta), \dots), \theta), \theta), \theta).$$

- In classical statistics the f_i are linear; but in ML it is necessary to use some kind of *non linarity*.

2.3 Common Layers

2.3.1 Fully Connected (Perceptron)

- Input: $X_i = (x_i^1, \dots, x_i^m)$.
- Parameters: $w_i j \in \mathbb{R}$ with $j = 1, \dots, m$ and bias $c \in \mathbb{R}^n$

- Fully Connected: Simply acts linearly:

$$f(X_i, W) = X_i W + c$$

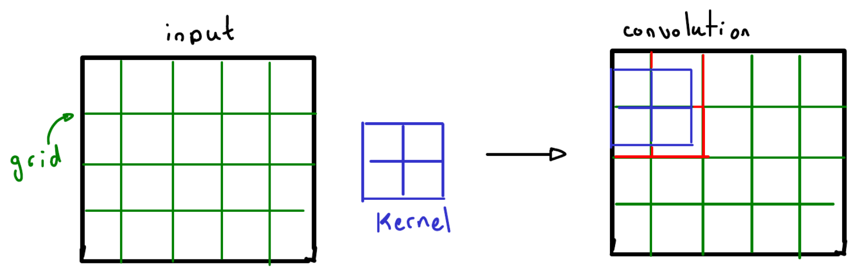
- Several limitations due to being linear.

2.3.2 Activation

- Using a non linear function to *increase* the representation power of the FC layers.
- Commonly used:
  - (Sigmoid) $g(t) = \frac{1}{1+e^{-\alpha t}}$
  - (Tan) $g(t) = \tanh(t) = \frac{e^t - e^{-t}}{e^t + e^{-t}}$
  - (Relu) $g(t) = \begin{cases} 0 & t < 0 \\ t & t \geq 0 \end{cases}$

2.3.3 Convolutional

- Networks with localized linear operators with an underlying grid geometry.



2.3.4 Attention Networks

Working

2.4 Optimization

2.4.1 Model

- Once constructed the **network architecture** the problem is now find θ s.t.:

$$\min_{\theta} L(f(\cdot,\cdot),\theta,D)$$

- It is possible?
  - Statistical properties of D, bias.
  - Representation power of f.
  - Practical problem of finding the minimum.

2.4.2 SGD

1. Steepest Descent Method
   - Using gradient:

   $$\nabla_{\theta} L_f(\theta; D) = \Delta\theta.$$

   - The iterative method using update:

   $$\theta_{i+1} = \theta_i + \alpha \Delta\theta_i$$

   - α is called the learning rate.

2. Difficulties
   - ☐ Selecting an appropriate θ_0.
   - ☐ Non convexity of L_f.
   - ☐ Number of iterations needed (time consumed).
   - Complexity on computing L_f for large D (Batch - SGD).
   - Automatically compute $D_{\theta} L_f$. Automatic differentiation and back propagation.
   - Check: arXiv:1609.04747 [cs.LG]

2.4.3 Loss Function

- In order to use Batch optimization we need that:

$$L_f(\theta; D_1 \cup D_2) \approx L_f(\theta; D_1) + L_f(\theta; D_2).$$

- The loss function is the way to give an additional bias to our model.

1. Common loss.

    - MSE: $\frac{1}{N}\sum_{j=1}^{N} |f(x_j;\theta) - t_j|$
    - CCE: $h(\hat{t}_i, t_i) = -(t_i \log(\hat{t}_i) + (1 - t_i)\log(1 - \hat{t}_i)$
    - ? Others (Wasserstein, Regularization)

3 Elementary Model Implementation

3.1 Modeling Process

1. Study the problem. (Domain Knowledge)
2. Study data, including a splitting. (?)
3. Propose model:
    - $f(;\theta)$
    - $L_f()$
4. Train Model
5. Model Evaluation (?)

3.2 Data Split (Basic)

- Since the objective of an ML is generalization i.e. a good performance on new data, we split the available data:
    1. Train
    2. Validation
    3. Test
- [?] k-fold validations, stratification.

3.3 Model Evaluation Beyond L_f

- L_f might not capture all the desired characteristics of the model.
- Over-fit [?]
- Accuracy - Sensitivity.
- Domain knowledge evaluation.
- Ethical (Bias).

3.4 Hands On:

- Sample Notebook repo (private).
- Trained Model: drive.

4 ToDo

4.1 Representative Power of Networks

Better Understanding:

- [ ] "Approximation by superposition of sigmoidal functions" by Cybenko,
- [ ] "Approximation capabilities of multilayer feedforward newtorks" by Hornik
- [ ] "Representation of Deep Forward Networks" by Telgarsky
- [ ] "On the computational efficiency of training Neural Networks" by Livni, Shalev
- [ ] "Complexity Theory Limitations for learning DFNs"

4.2 Algorithms

- [ ] "Guaranteed Training of Neural Networks using Tensor Methods" by Janzamin
- [ ] "Train faster, generaliza better" by Hardt

4.3 Understanding Boltzman Machines

☐ "Probable Bounds for Learning Some Deep Representations" by Arora

☐ "Deep Learning and Generative Hierarchal models" by Mossel