Statistical Computing

Michael Mayer

March 2023

Statistical Computing: What will we do?

Chapters

1. R in Action
2. Statistical Inference
3. Linear Models
4. Model Selection and Validation
5. Trees
6. Neural Nets

Remarks

- Chapters 3 to 6:
  Statistical ML in Action
- Two weeks per chapter
- Exercises at end of chapter notes

Neural Nets

Outline

- Understanding Neural Nets
- Practical Considerations
- Extended examples

Neural Nets

- Around since the 1950ies
- Underwent different development steps, e.g.
  - use of backpropagation (Werbos, 1974) use chainrule to use more hidden layer
  - GPUs (2009, ImageNet 2012) first win by deep learning by far
- Black Box more than boosted trees
- TensorFlow/Keras, PyTorch

Google wrapper around TF facebook

“Swiss Army Knife” among ML Algorithms

like special case of NN

Can fit linear models

Fit data larger than RAM

Learn interactions and non-linear terms

Non-linear dimension reduction

PCA

Learn «online»

production model running + new data learned on the fly

>1 Responses possible

Sequential and spatial in- and output

Flexible and mixed in- and output dimensions

model that takes numerical, text and image of house → output price

Flexible loss functions

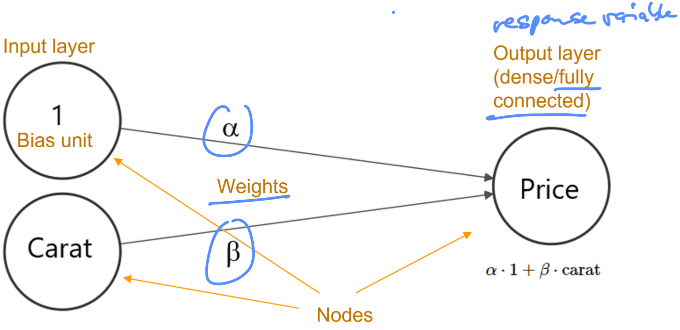
Understanding Neural Nets in three Steps

1. Linear regression as neural net
2. Hidden layers
3. Activation functions

Using diamonds data

Step 1: Linear Regression as Neural Net

- $\mathbb{E}(\text{price}) = \alpha + \beta \cdot \text{carat}$
- OLS
  $\hat{\alpha} \approx -2256$, $\hat{\beta} \approx 7756$
- Represented as neural network graph



Example

The Optimization Algorithm

Mini-batch gradient descent with backpropagation

Notation: Neural net f_β; its total loss on data D and loss function L:

$\beta = \binom{\alpha}{\beta}$

$$Q(f_\beta, D) = \sum_{(y_i, \boldsymbol{x}_i) \in D} L(y_i, f_\beta(\boldsymbol{x}_i))$$

boosted trees
are convex
→ converges everytime

1. Init: Randomly initialize parameter vector β by $\hat{\beta}$
2. Forward: Calculate $Q(f_{\hat{\beta}}, D_{\text{batch}})$ on batch
3. Backprop: Modify $\hat{\beta}$ to improve $Q(f_{\hat{\beta}}, D_{\text{batch}})$

   chain rule

   3.1 Calculate partial derivatives $\nabla\hat{\beta} = \frac{\partial Q(f_\beta, D_{\text{batch}})}{\partial \beta} |_{\beta=\hat{\beta}}$ using backprop (=?)
   3.2 Gradient descent: Move slightly into right direction: $\hat{\beta} \leftarrow \hat{\beta} - \lambda \cdot \nabla\hat{\beta}$
4. Repeat Steps 2 and 3 until one epoch is over

   you have touched each datapoint

5. Repeat Step 4 until some stopping criterion triggers

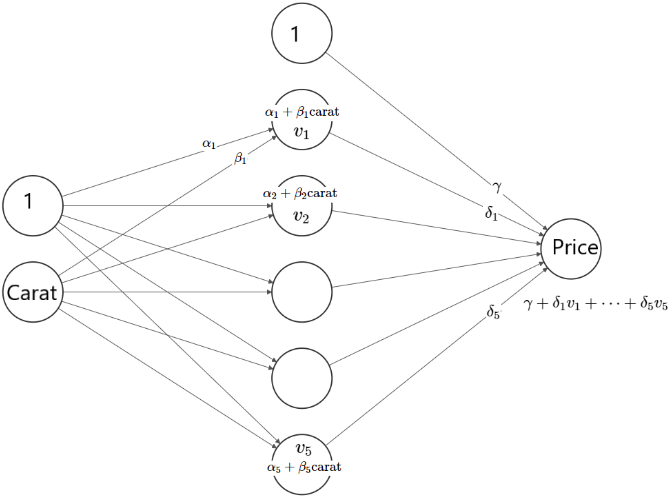
SGD? Local minima?

for us: make a simple model to compare to

Step 2: Hidden Layers

- Add hidden layers for more parameters (= flexibility)
- Their nodes are latent/implicit variables
- Representational learning
- Encoding? / Embedding
- Deep neural net?

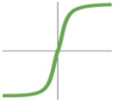
Example



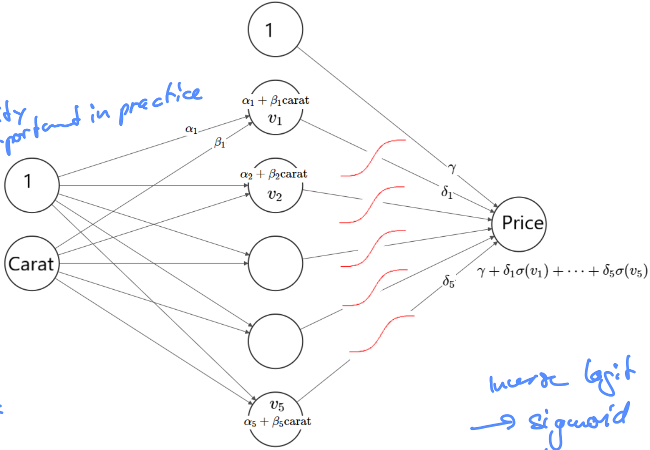
Step 3: Activation Functions

Non-linear transformations σ of node values necessary!

- tanh: $\sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- ReLU: $\sigma(x) = \max(0, x)$



discontinuity is not important in practice



Two purposes

- Imply interactions and non-linear terms
- Inverse link as in GLMs

activation = inverse link

Example

→ GLM

inverse logit

→ sigmoid $\frac{1}{1+e^{-x}}$

Practical Considerations

no cross validation, b/c nn take long to fit

Validation and tuning of main parameters

important initial guess good, if input is standardized

Input standardization

Missing values

Categorical input

dummy / integer encoding / embedding layer

Callbacks

early stopping

Types of layers

Interpretation

Optimizer

try to not be stuck in local minima

Overfitting and regularization

50-100 rows / parameter

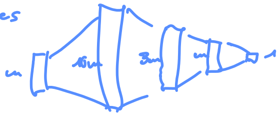
Choosing the architecture

for tabular data: m features

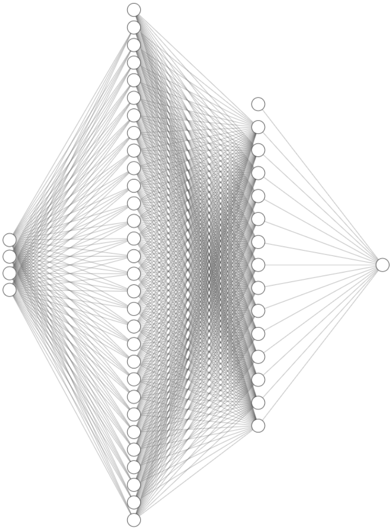
1.) 10·m hidden nodes
→ learn interactions

2.) 3m , 3.) m 4) 1

Custom losses and evaluation metrics



Example: Diamonds



Excursion: Model-Agnostic Importance Measure

no native way to measure variable importance

Permutation importance of feature $X^{(j)}$, data D, and performance measure S:

$$\mathrm{PVI}(j, D) = \overbrace{S(\hat{f}, D^{(j)})}^{\text{shuffled}} - \underbrace{S(\hat{f}, D)}_{\text{e.g. RMSE on validation / test data}}$$

- $D^{(j)}$ is version of D with randomly permuted values in j-th feature column
- Read: How much S worsens after shuffling column j?
  The larger, the more important. If 0, feature is unimportant
- Computationally cheap $\rightarrow$ repeat m times
- Model is never refitted
- Training or test data?
  on test data

Example

Embeddings

Represent unordered categorical X with K levels by $m << K$ numeric features

Embedding layer

- X integer encoded
- Dummy matrix $\tilde{X}$ with K columns
- Multiply $\tilde{X}$ with $(K \times m)$ matrix β
- Embedding matrix β estimated like other parameters
- Trick: $\tilde{X}\beta$ is calculated via index slicing from X and β
  $\to \tilde{X}$ is never materialized
- Think: $X_1 = j \to$ first row of $X\beta$ equals j-th row of β etc.

Example

Taxi trips

Excursion: Analysis Scheme X

$T(Y)$: quantity of interest

Steps

1. Calculate $T(Y)$ on the full data
2. Calculate $T(Y)$ stratified by covariates $X^{(j)}$ $\rightarrow$ bivariate associations
3. Accompany Step 2 by ML model $\rightarrow$ multivariate associations
   - Study model performance
   - Study variable importance $\rightarrow$ sort results of Step 2
   - Study PDP (or similar) for each $X^{(j)}$ and compare with Step 2

Example

Comparison of ML Algorithms

Aspect	GLM	Neural Net	Decision Tree	Boosting	Random Forest	k-Nearest Neighbour
Scalable	😍	😍	😀	😀	😐	😞
Easy to tune	😐	😐	😐	😐	😀	😐
Flexible losses	😀	😍	😀	😀	😐	😐
Regularization	✔️	✔️	✔️	✔️	✔️	✔️
Case weights	✔️	✔️	✔️	✔️	✔️	✔️
Missing input allowed	😞	😞	✔️	✔️	😞	😞
Interpretation	😍	😐	😍	😐	😐	😐
Space on disk	😍	😍	😍	😀	😞	😞
Birth date (approx.)	1972 (Nelder & Wedderburn)	1974 Backprop (Werbos)	1984 (Breiman et al.)	1990 (Schapire)	2001 (Breiman)	1951 (Fix & Hodges)