Datorlaboration 3

Josef Wilzén

August 31, 2020

732G12 Data Mining HT2020

Allmänt

Datorlaborationerna kräver att ni har R och Rstudio installerat.

- Kodmanual: länk
- Dataset till vissa uppgifter finns här.
- **ISL**: An Introduction to Statistical Learning,
  - Allmän info: länk
  - Boken: länk
  - R-kod till labbar: länk
  - Dataset: länk

Notera att ni inte behöver göra alla delar på alla uppgifter. Det viktiga är att ni får en förståelse för de olika principerna och modellerna som avhandlats. Dessa uppgifter ska inte lämnas in, utan är till för er övning.

Datauppdelning

För att motverka överanpassning bör ni dela upp data till träning-, validering-, (och testmängd). Detta kan göras med `createDataPartition()` från `caret`-paketet. Argument till den funktionen som är av vikt här är p som hur stor andel av observationerna som ska användas till träningsmängden. Ni kan också använda `subset()` för att göra detta också, men det blir svårare att tydligt ange de observationer som ska tilldelas till valideringsmängden. Denna uppdelning ska ske slumpmässigt. Notera att om en testmängd ska skapas måste uppdelningen ske en gång till från valideringsmängden.

Del 1: Installera Keras

- Installera keras-paketet, se kodmanualen för instuktioner.
- Dokumentation för keras finns här

Del 2: Neurala Nätverk: klassificering

Målet med denna övning är att se hur bra neurala nätverk är på att känna igen bilder med siffror.

1. Läs in materialet "mnist_train.csv" som är träningsmängden från MNIST databasen och ange rätt skala på variablerna. Här lämpar sig `read.csv2()`. Läs också in testmängden "mnist_test.csv" . Tänk på hur keras-paketet vill ha data strukturerat och följ instruktionerna i kodmanualen.

2. Skatta ett neuralt nätverk med ett gömt lager och 10 gömda neuroner. Aktiveringsfunktionen ska vara Relu i det gömda lagret och Softmax i outputlagret. I träningsfasen ska ni ni ta bort den så kallade interna validering, alltså att träningsmängden delas upp till en valideringsmängd inuti algoritmen. Detta görs genom att ange `validation_split = 0`. Använd de övriga standardvärden som är angivna i kodmanualen för anpassning. Ta reda på hur många parametrar er modell har.

3. Utvärdera modellen som skattats genom att kontrollera nätverkets anpassningshistorik. Hur många epoker behövdes för att hitta den bästa modellen? Verkar modellen vara nog bra för att modellera data- materialet? Utifrån träningsmängden vilka siffror verkar modellen ha sämst lycka med att prediktera? Vilka siffror har modellen predikterat de till istället?

4. Utvärdera även modellen på er testmängd. Hur ser resultatet ut där?

5. Testa nu att anpassa modellen med intern validering (20 procent) och repetera steg 3) och 4). Hur ser resultatet ut nu? Kan man teoretiskt förvänta sig en bättre modell med denna förändring?

6. Testa nu att skatta en modell med intern validering där ni försöker ändra arkitekturen av nätverket, t.ex. aktiveringsfunktionerna, antalet gömda neuroner, antalet gömda lager osv. Vad verkar producera bättre resultat?

7. Testa nu att skatta en modell där ni försöker ändra optimeringen av nätverket, t.ex. antalet epoker, batchstorleken, learning rate, utvärderingsmåttet osv. Vad verkar producera bättre resultat?

8. Sammanfatta alla dessa modeller som ni skattat och dra slutsatser om huruvida denna sorts data är lämplig att skatta med neurala nätverk och vad som krävs för att få bra resultat.

9. Gå igenom denna tutorial: länk. Här används också MNIST-data, men data läses in med funktionen dataset_mnist(), notera att data har annat format än när ni läste in den i 1), nu är det en array. Dett gör att

inputlagret behöver vara annorlunda. Anpassa den föreslagna modellen. Hur presterar den jämfört med er tidigare? Hur många parameterar skattar ni? Förutom själva arkitekturen på nätverket, hittar ni några andra skillnader jämfört med era tidgare modeller?

Del 2: Mer klassificering

Kör denna tutorial: länk.

1. Vad är det för sorts data ni jobbar med här? Hur många obs? Hur stora bilder?
2. Skatta den föreslagna modellen. Hur många parameterar har den?
3. Hur presterar modellen på testdata? Beräkna förväxlingsmatrisen för testdata.
   (a) Vilken klass var lättast att klassificera?
   (b) Vilken klass var svårat?
4. Testa att ändra arkitekturen på modellen.
   (a) Ändra antalet neuroner i lagren
   (b) Testa att lägga till fler lager.
5. Testa att ändra optimeringen
   (a) antalet epoker
   (b) batchstorlekek
6. Hur presterar den bästa modellen som ni lyckas hitta?
7. Se denna video: Parameters vs Hyperparameters

Del 3: Neurala Nätverk: Regression

Kör denna tutorial: länk.

1. Vad är det för data? Hur många förklarande variabler? Hur många observationer?
2. Skatta den föreslagna modellen. Hur många parameterar har den?
   (a) Hur bra blir modellen på testdata?
   (b) Vilken kostnadsfunktion används?
3. Testa att ändra arkitekturen på modellen och utvärdera resultatet på testdata.
4. Testa att ändra optimeringen och utvärdera resultatet på testdata.
5. Skatta en Random forest-modell på samma dataset och utvärdera resultatet på testdata.
   (a) Fungerar Random forest bättre eller sämre än neurala nätverk på dettta dataset?

Del 4: Mer regression

Ni ska nu modellera datasetet det simulerade datasetet "NN_reg_data.csv" med neurala nätverk. Datasetet har en prediktor, x och y har ett icke-linjärt samband med y.

1. Läs in datamaterialet i R.
2. Det finns tre olika y-variabler:
   - `y_true`: det sanna värdet på y
   - `y_less_noise`/`y_more_noise`: `y_true` + olika nivåer av brus.
3. Plotta de olika y mot x. Hur ser sambandet ut?
4. Börja med att ha y_less_noise som er responsvariabel.
5. Dela upp data i träning/test (80/20) slumpmässigt
   (a) Skatta[1] neurala nätverk med följande arkitekturer:
      i. Ett gömt lager med 5 noder + Relu
      ii. Ett gömt lager med 30 noder + Relu
      iii. Ett gömt lager med 100 noder + Relu
      iv. Två gömda lager med 20 + 20 noder +Relu
   (b) Vilken modell presterar bäst på testdata?
   (c) Plotta anpassade värden för träning/test i samma plot som `y_true`. Ser det ut att vara en bra anpassning?
   (d) Om ni vill: Skatta random forest och k-nearest neighbors och jämför vilken modell som presterar bäst på testdata.
6. Dela upp data i träning/test, där de 20% sista observationerna är testmängden (alltså de längst åt höger i era plottar) och resterande i träningsdata.
   (a) Skatta[2] neurala nätverk med följande arkitekturer:
      i. Ett gömt lager med 5 noder + Relu
      ii. Ett gömt lager med 30 noder + Relu
      iii. Ett gömt lager med 100 noder + Relu
      iv. Två gömda lager med 20 + 20 noder +Relu
   (b) Vilken modell presterar bäst på testdata?
   (c) Plotta anpassade värden för träning/test i samma plot som `y_true`. Ser det ut att vara en bra anpassning?

[1] Välj valfria inställningar på optimeringen.
[2] Välj valfria inställningar på optimeringen.

(d) Om ni vill: Skatta random forest och k-nearest neighbors och jämför vilken modell som presterar bäst på testdata.

7. I 5) försöker ni interpolera en funktion, i 6) försöker ni extrapolera en funktion. Vilken uppgift verkar lättast? Hur fungerade era modeller i de båda fallen?

8. Upprepa 5) och 6) fast använd `y_more_noise`. Hur presterar de olika modellerna när det finns mer burs i data?

9. Skatta några modeller på `y_true` och hela datasetet som träning.

   (a) Skatta modeller med ett gömt lager och:

      i. 5 noder + Relu

      ii. 10 noder + Relu

      iii. Så många noder som krväs för att få en mycket bra anpassning.

   (b) Plotta observerade värden och anpassade värden i samma plot. Hur bra är neurala nätverk på att anpassa en godtycklig funktion?