Oppgaver Løsningsforslag - SOK3023

Maskinlæring for økonomer
Laget av Markus J. Aase

January 29, 2025

Dette dokumentet inneholder oppgaver studentene i SOK-3023 kan jobbe med etter andre uka av kurset. En del av svarene vil dere kunne finne fra de fysiske forelesningene og/eller kompendiet. Mens noen spørsmål krever at dere oppsøker informasjonen selv i dokumentasjon til Tensorflow eller andre sted på internett.

1. **Bias-Variance Trade-off**

   Forklar bias-variance trade-off og beskriv hva som skjer med modellens ytelse dersom bias er for høy eller variansen er for høy.

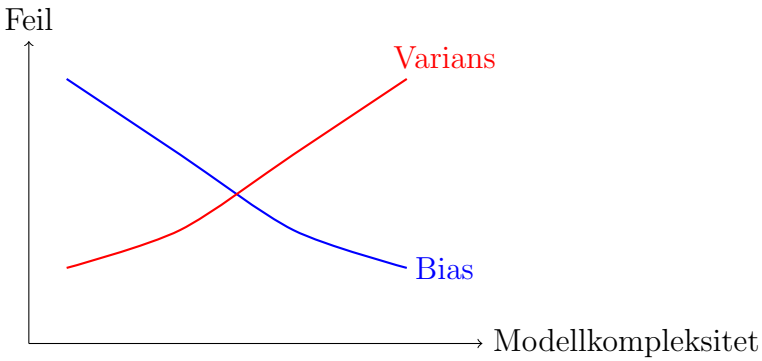
   Løsning:

   Bias-variance trade-off handler om å balansere mellom to typer feil:

   - **Høy bias** betyr at modellen er for enkel (underfitting). Den fanger ikke opp mønstre i dataene og gir høy feil både på trenings- og testdata.

   - **Høy varians** betyr at modellen er for kompleks (overfitting). Den tilpasser seg treningsdataene for godt og presterer dårlig på nye, usette data.

   Dette kan ofte illustres slik, hvor vi ønsker å finne en balanse mellom bias og varians, for å unngå overfitting og underfitting.



   NB: Merk dere at dette er et kort, enkelt svar og at det ligger mer bak her for den interesserte!

2. **Veiledet vs. ikke-veiledet læring**

   Hva er forskjellen mellom veiledet og ikke-veiledet læring? Gi eksempler på hver type læring.

   Løsning:

   - **Veiledet læring**: Modellen trenes på data med kjente utfall (labels).

   - Eksempel: E-postklassifisering (spam/ikke spam).

   - **Ikke-veiledet læring**: Modellen oppdager mønstre i data uten labels.

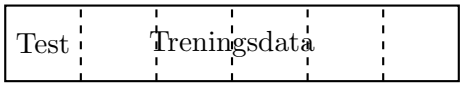
   - Eksempel: Klynging av kunder basert på kjøpshistorikk.

3. **Kryssvalidering**

   Hva er kryssvalidering, og hvorfor brukes det? Beskriv en vanlig metode for kryssvalidering.

   Løsning:

   Kryssvalidering brukes for å vurdere en modells generaliseringsevne ved å dele datasettet i trenings- og testsett flere ganger. En vanlig metode er **k-fold kryssvalidering**, der datasettet deles i k like store deler. Hver del brukes som testsett én gang, mens de resterende $k-1$ delene brukes som treningssett.



4. **Evaluering av regresjonsmodeller**

   Sammenlign Mean Squared Error (MSE) og Mean Absolute Error (MAE). Hva er fordelene og ulempene med hver metode?

   F.eks. hvordan takler de *outliers* (altså hvor det er stor forskjell mellom $\hat{y}_i$ (predikert) og y_i (observert).

   Formler:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2, \quad \text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|.$$

5. **Evaluering av klassifikasjonsmodeller**

   Forklar med egne ord hva accuracy, precision og recall er, gjerne bruk eksempelet under. Bruk følgende formler:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN},$$

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall/Sensitivitet} = \frac{TP}{TP + FN}.$$

(a) Anta at vi har en modell som forsøker å identifisere kredittkortsvindel. Modellen klassifiserer transaksjoner som enten **ikke-svindel (0)** eller **svindel (1)**. Vi evaluerer modellen ved å bruke en **confusion matrix** basert på 1000 testtransaksjoner:

	Predikert: Ikke-svindel (0)	Predikert: Svindel (1)
Faktisk: Ikke-svindel (0)	**950** (TN)	**30** (FP)
Faktisk: Svindel (1)	**15** (FN)	**5** (TP)

Er dette en god modell? Hvorfor/hvorfor ikke?

- **True Negatives (TN) = 950**
  Modellen korrekt identifiserer mange ikke-svindeltransaksjoner.
- **False Positives (FP) = 30**
  Modellen feilaktig markerer noen ikke-svindeltransaksjoner som svindel.
- **False Negatives (FN) = 15**
  Modellen overser 15 tilfeller av faktisk svindel.
- **True Positives (TP) = 5**
  Modellen korrekt identifiserer kun 5 svindeltilfeller.

Løsning:

- **Accuracy**: Hvor stor andel av alle prediksjonene var riktige.

- **Precision**: Hvor stor andel av de positive prediksjonene var riktige.

- **Recall**: Hvor stor andel av de faktiske positive tilfellene ble korrekt identifisert.

For svindel/ikke-svindel confusion matrix'en, kan vi regne ut disse verdiene å si noe om modellen. Det klarer dere selv!:)

6. **MNIST-datasettet**

   Hvordan kan et tall fra MNIST-datasettet gjøres om til inputlaget i et nevralt nettverk? Illustrer.

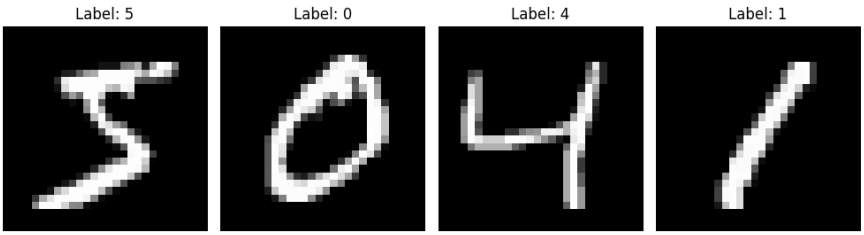


Figure 1: Eksempel på MNIST-data, som brukes til håndskriftgjenkjenning i maskinlæring.

Løsning: Hvert bilde i MNIST består av 28×28 piksler som kan representeres som en vektor på 784 verdier - og derfor som en input lag i et MLP.

7. **Nevrale nettverk**

   Forklar arkitekturen i et nevralt nettverk. Beskriv input-laget, skjulte lag og output-laget. Illustrer.

   Løsning:

   Et nevralt nettverk består av:

   - **Input-lag**: Tar imot input-data (f.eks. piksler som en lang vektor).

   - **Skjulte lag**: Lærer mønstre ved å bruke aktiveringsfunksjoner.

   - **Output-lag**: Gir en prediksjon (f.eks. 0 eller 1 for binær klassifikasjon).

8. **Aktiveringsfunksjoner**

   Hva er en aktiveringsfunksjon? Gi tre eksempler, forklar hvordan de fungerer og hvorfor de er relevant i nevrale nettverk.

   Løsning:

   En aktiveringsfunksjon er en matematisk funksjon som brukes i nevrale nettverk for å bestemme om ett nevron skal aktiveres eller ikke. Funksjonen tar inn en verdi og returnerer et output, som ofte brukes til å introdusere ikke-linearitet i modellen. Dette gjør at nevrale nettverk kan lære mer komplekse mønstre.

   Tre eksempler på aktiveringsfunksjoner:

   (a) **Sigmoid**: Sigmoid-funksjonen er en av de mest brukte aktiveringsfunksjonene. Den gir en kontinuerlig utgang mellom 0 og 1. Formelen for sigmoid er:

   $$f(x) = \frac{1}{1 + e^{-x}}$$

   Sigmoid-funksjonen er spesielt nyttig for problemer der vi ønsker sannsynligheter (for eksempel i binær klassifisering).

   (b) **ReLU (Rectified Linear Unit)**: ReLU er en enkel og populær aktiveringsfunksjon som returnerer x hvis $x > 0$, og 0 ellers:

   $$f(x) = \max(0, x)$$

   ReLU er kjent for å bidra til raskere konvergens under trening, og er ofte brukt i dype nevrale nettverk.

   (c) **Tanh (Hyperbolic Tangent)**: Tanh-funksjonen er en annen aktiveringsfunksjon som returnerer en verdi mellom -1 og 1. Formelen for tanh er:

   $$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

   Tanh er nyttig når vi trenger en funksjon som har både positive og negative verdier, og den er ofte brukt i rekursive nevrale nettverk (RNN), som vi vil se bittelitt på helt i slutten av pensum.

9. **Hva betyr *læring* i maskinlæring?**

   Forklar hva *læring* refererer til i maskinlæring, og hvordan en modell forbedrer seg over tid.

   Løsning:

   I maskinlæring refererer *læring* til prosessen hvordan en modell forbedrer sine prediksjoner eller beslutninger ved å analysere data. Modellen lærer å generalisere fra treningsdataene slik at den kan gjøre riktige prediksjoner på nye, usette data.

   Læring skjer vanligvis gjennom iterativ oppdatering av modellens parametere (for eksempel vekter i et nevralt nettverk) basert på feilen mellom modellens prediksjoner og de faktiske resultatene. Denne prosessen kan innebære metoder som gradient descent, som justerer modellens parametere for å minimere feilen. (Gradient decent og backpropagation er viktige prinsipper her, og det skal vi diskutere mer i neste uke!!)

   Maskinlæring kan deles inn i forskjellige typer:

   - **Veiledet læring (supervised learning)**: Modellen lærer fra et datasett som inneholder både input og de riktige outputene (merkelapper, ofte kalt *labels*). Dette er fokuset i dette emnet.
   - **Ikke-veiledet læring (unsupervised learning)**: Modellen lærer mønstre i dataene uten å ha tilgang til labels.
   - **Forsterkende læring (reinforcement learning)**: Modellen lærer gjennom interaksjon med miljøet og får belønning eller straff basert på handlingene den tar. Ikke en del av dette kurset.

10. **Batch, epoch og loss-funksjon**

    Forklar forskjellen mellom batch, epoch og loss-funksjon i maskinlæring. Hvordan påvirker disse modelltreningen?

    Løsning:

    I maskinlæring refererer *batch*, *epoch* og *loss-funksjon* til viktige konsepter i trening av modeller.

    - **Batch**: En batch er en delmengde av treningsdataene som brukes til å oppdatere modellens parametere i én iterasjon. Ved å bruke batcher i stedet for hele treningssettet, kan vi gjøre beregningene mer effektive og samtidig unngå minnebegrensninger.

      Tenk på et treningssett som består av 1000 datapunkter. Hvis vi oppdaterte modellens parametere etter å ha sett alle 1000 datapunktene, ville det vært batch gradient descent (også kalt, full batch). Men ofte deler vi treningssettet opp i mindre batcher for å gjøre læringen mer effektiv. La oss si vi velger en batchstørrelse på 100. Dette betyr at vi deler opp treningsdataene i 10 batcher (100 datapunkter per batch).

- **Epoch**: En epoch er en full gjennomgang av treningsdataene. Etter hver epoch har modellen sett hele treningssettet én gang, og dens parametere oppdateres basert på dataene i batchene. Modellen blir vanligvis trent over flere epochs for å forbedre ytelsen.
- **Loss-funksjon**: En loss-funksjon (eller tapsfunksjon) er en matematisk funksjon som kvantifiserer hvor godt eller dårlig en modell presterer. Den beregner feilen mellom modellens prediksjoner og de faktiske resultatene. Målet under trening er å minimere verdien av loss-funksjonen. Eksempler på loss-funksjoner er:
  - **Mean Squared Error (MSE)**: Brukes ofte i regresjonsproblemer, og beregner gjennomsnittlig kvadratisk feil mellom prediksjonene og de faktiske verdiene.
  - **Cross-Entropy Loss**: Brukes ofte i klassifiseringsproblemer, spesielt når man jobber med sannsynligheter for ulike klasser.

Disse begrepene er avgjørende for modellens treningsprosesser og bestemmer hvor effektivt og nøyaktig modellen lærer fra dataene den blir trent på.

NB: For at man skal ha en god maskinlæringsmodell må man velge riktig loss-funksjon. TensorFlow har god dokumentasjon på dette.