Deep Learning

2023-11-29

Ämne: Databehandling, Regulariseringstekniker, Hyperparameteroptimering

Smartr

Agenda

- Databehandling
  - Onehot encoding
  - Normalisering + Standardisering
  - Feature engineering
  - Obalanserad data
  - Data augmentering
- Regulariseringstekniker
  - Early stopping
  - L1/L2 Regularisering
  - Drop out
- Hyperparameteroptimering

Databehandling

- Data är a och o för deep learning.
- Generellt gäller mer träningsdata -> bättre resultat
  - Diminishing returns dock.

- Men det hjälper även modellen om
  - Datan är av hög kvalité och städad.
  - Dataseten är balanserade.
  - Man representerar datan smart genom *feature engineering.*

Städa data

- Ta bort duplicerade datapunkter
- Se till att datan är formaterad lika (t.ex. ”N/A” och ”Not applicable” och ”-” är samma sak)
- Eventuellt filtrera bort oönskade outliers
- Hantera saknad data
  - Kan du rimligt gissa vilket värde borde vara där?
  - Eller bör du ta bort datapunkten?
- Validera datan
  - För varje kolumn (feature) ta reda på max, min, mean, std…
  - Är förväntad unik data faktiskt unik?
  - Är datan rimlig? (Baseras på domänkuskap)

Encoding

- Kategorisk data är svårt att mata in direkt i en ML modell.
- Måste representeras på ett numeriskt sätt.

- Finns olika encoding som är vanliga:
  - Ordinal encoding
  - Dummy encoding
  - One-hot encoding

Ordinal Encoding

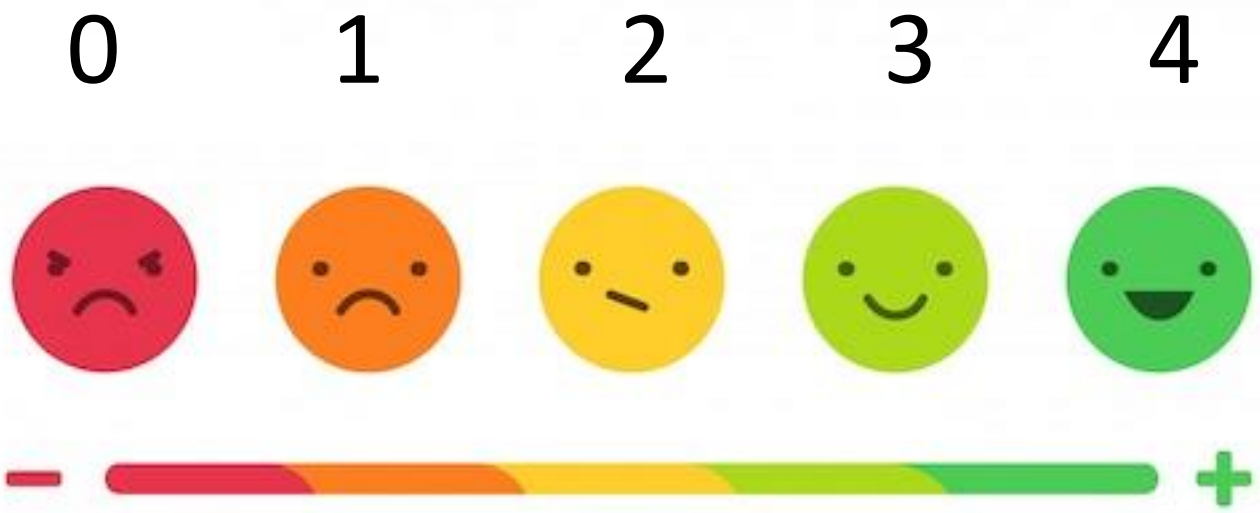
- Vi har färgerna grön, lila, gul och röd som vi ska encodea.

- Ordinal encoding ger dem olika värden i samma feature.
- 0=grön, 1=lila, gul=2, röd=3.

- Detta blir ganska dåligt i detta exempel eftersom grön är närmare lila än röd i feature-rymden vi skapat.

Ordinal Encoding

Ordinal encoding kan vara bra när det finns en inneboende ordning. T.ex. nöjdhet från en enkät.



One-hot Encoding

- Vi har frukterna äpple, plommon, citron och tomat som vi ska encodea.
- One-hot encoding gör att varje frukt blir en egen feature som är 0 eller 1.
- Om datapunkten är kategorin äpple blir feature vektorn [1, 0, 0, 0]
- Det blir många features (dimensioner i input datan) men de är helt separerade. Det finns ingen ordning eller koppling mellan dem.

Dummy Encoding

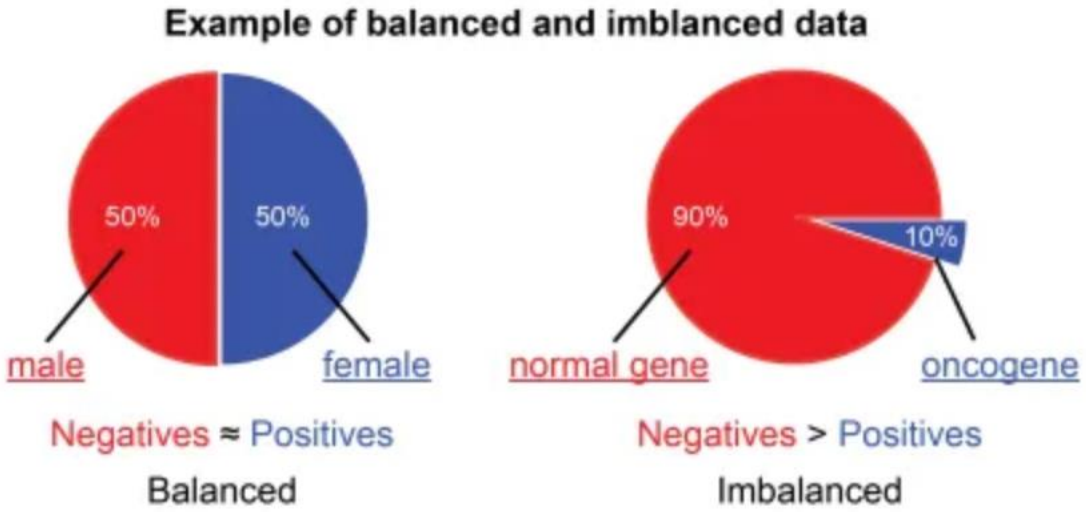
- Dummy encoding och one-hot encoding är väldigt lika.
  - One-hot encoding skapar en vektor med **N** features.
  - Dummy encoding skapar en vektor med **N-1** features.
- Den sista klassen representeras av att alla features sätts till 0.

Frukt (kategori)	One-hot encoding	Dummy encoding
Äpple	[1, 0, 0, 0]	[1, 0, 0]
Plommon	[0, 1, 0, 0]	[0, 1, 0]
Citron	[0, 0, 1, 0]	[0, 0, 1]
Tomat	[0, 0, 0, 1]	[0, 0, 0]

One-hot encoding oftast bäst för deep learning

Balansera dataset

- Varför är det ett problem att ett dataset som ni ska använda för träning är obalanserat?



Balansera dataset

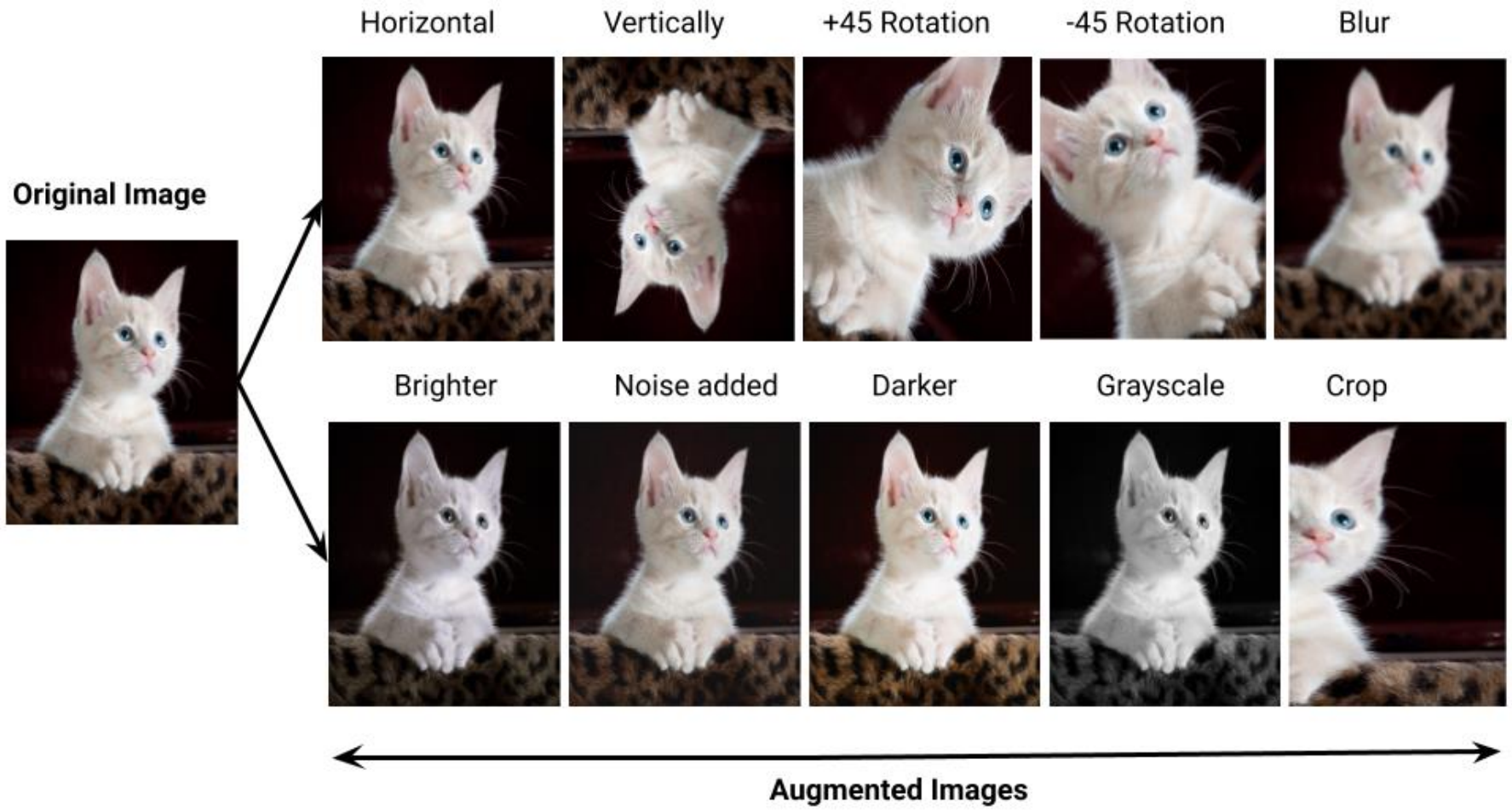
- Hur balanserar man ett dataset?

- Skaffa mer data av det du har för lite av.
  - Kan vara svårt och tidskrävande
  - I vissa fall kan data augmentering eller data generering funka.
- Använd inte allt av datan du har för mkt av.
  - Sampla data balanserat vid träning.

Data augmentering

- Vi kan artificiellt utöka datasetet genom att augmentera datat.
- Detta funkar väldigt bra för data som består av bilder.
  - Att augmentera annan data kan vara svårt för det kan vara svårt att bedöma att innehållet ändå fortfarande stämmer med ground truth. För en bild är det enkelt att se att, t.ex. det fortfarande är en katt.
- Exempelvis kan vi göra olika kombinationer av:
  - Shift, shear, scale, rotate, ändra färg.

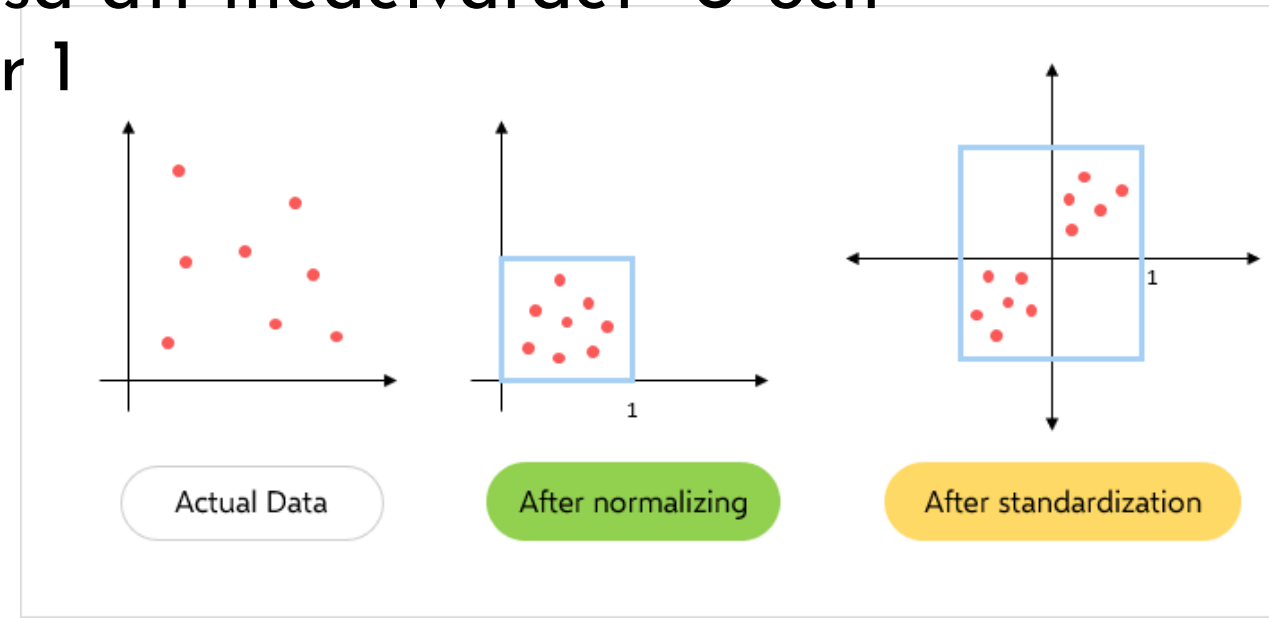
Data augmentering - exempel



Normalisering & standardisering

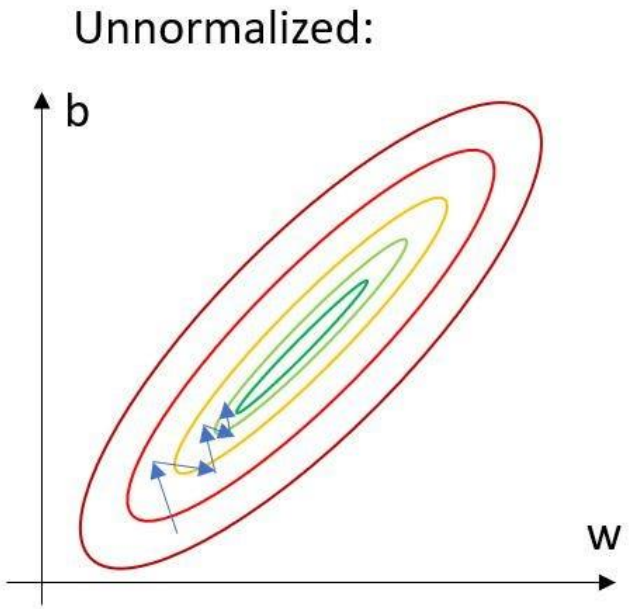
- Normalisering innebär att skala om en feature så max=1 och min=0 (eller max=1 och min=-1 om negativa värden).

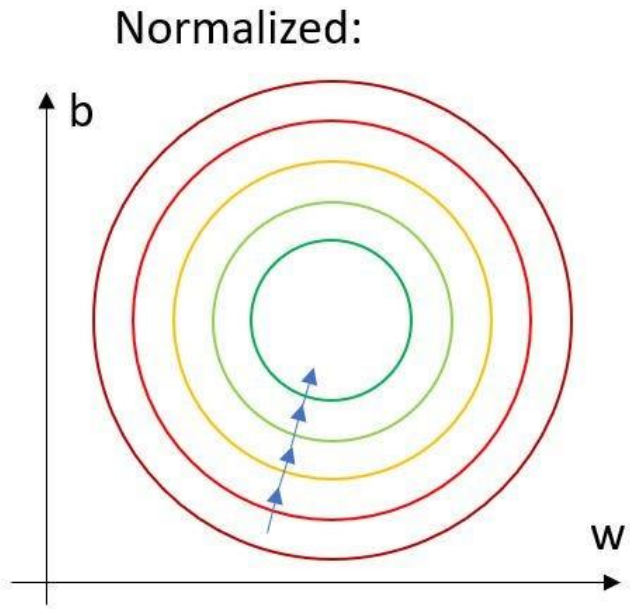
- Vid standardisering antar man att datan är normalfördelad och skalar om datan så att medelvärdet=0 och standardavvikelsen är 1



Normalisering – intuition varför hjälper det

- Normalisering hjälper vid träning av neurala nätverk eftersom alla features håller samma skala.
- Detta stabiliserar gradient decsent och gör att modellen konvergerar lättare.





Feature engineering

- Man kan inte smälla på normalisering eller one-hot encoding på måfå utan du behöver reflektera över vad datan betyder och hur den sitter ihop.

Feature engineering exempel

- Hur skulle ni representera ett klockslag som feature? HH:mm

Feature engineering exempel

- Hur skulle ni representera ett klockslag som feature? HH:mm

- One-hot encoding
  - Inte bra då det inte riktigt är klasser (och blir väldigt många klasser).
  - One-hot encoding antar att klasserna inte har någon inneboende koppling till varandra men kl 14:00 är närmare 13:59 än 14:10.

- Normalisering / standardisering av reellt värde.
  - Bättre, men inte helt bra då kl 23:59 och 00:00 hamnar längst ifrån varandra i feature rymden, men de är egentligen väldigt nära. Vill man hitta mönster i datan som beror på tid på dygnet tappar man den cirkulära informationen.

Feature engineering exempel

- Hur skulle ni representera ett klockslag som feature? HH:mm

- Vi kan representera det som två kordinater i en enhetscikel!
- Cirkulära beroendet behålls.

Regulariseringstekniker

- Tidigare har vi pratat om early stopping

- Varför använder man regularisering?
  - Undvika overfitting!

L1 och L2 regularisering

- L1 och L2 regularisering är en term man lägger till på loss-funktionen.

Cost function = Loss(...) + regulariserings term

Eftersom detta sen propageras genom backpropagation kommer denna termen påverka matrisen av vikter. Genom termen blir viktmatrisen bestraffad av att vara stor och komplex och tvingas att bli mindre och enklare.

Rent krasst så minskar detta overfitting eftersom nätverket tvingas hitta enklare mönster som stämmer på flera olika exempel -> generaliserar bättre.

L2 regularisering (Weight decay)

- L2 regularisering kallas även weight decay eller ridge regression

$$Cost\,function = Loss + \frac{\lambda}{2m} * \sum \|w\|^2$$

lamda = en hyperparameter, m = batch size, w = viktmatrisen

- Denna termen tvingar vikterna att gå mot 0, (men inte bli 0).

L1 regularisering (Lasso)

- L1 regularisering kallas även Lasso regularization.

$$Cost\,function = Loss + \frac{\lambda}{2m} * \sum ||w||$$

lamda = en hyperparameter, m = batch size, w = viktmatrisen

- Denna kan reducera vikterna till 0, vilket är användbar om man vill komprimera sin modell då det ”tar bort” features. Oftast föredras L2, då behålls samtliga features men får olika inflytande

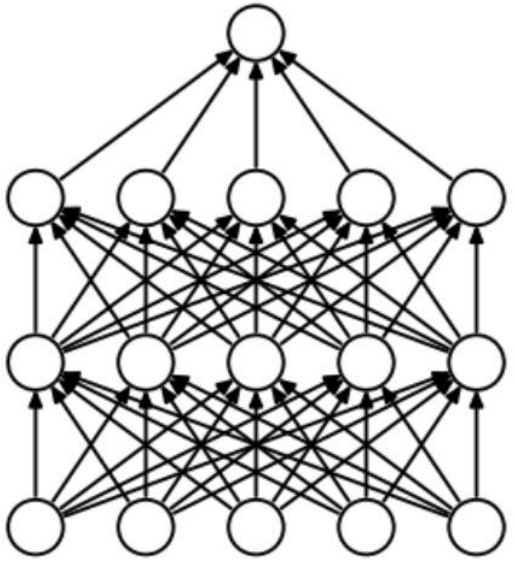
L1 vs L2 (weight decay)

L1 Regularization	L2 Regularization
1. L1 penalizes sum of absolute values of weights.	1. L2 penalizes sum of square values of weights.
2. L1 generates model that is simple and interpretable.	2. L2 regularization is able to learn complex data patterns.
3. L1 is robust to outliers.	3. L2 is not robust to outliers.

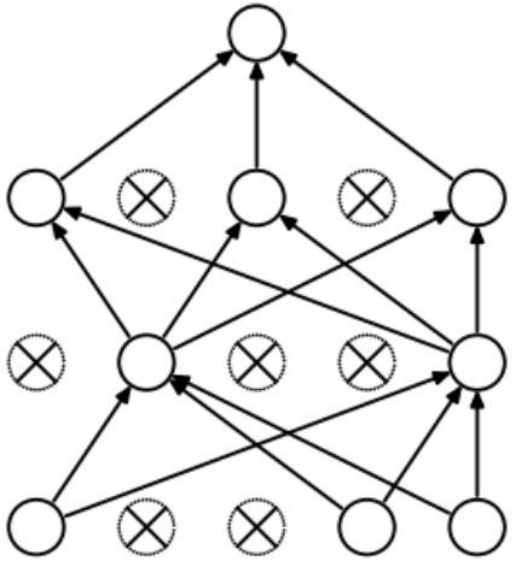
Drop out

- Drop out är en vanlig och effektiv metod för regularisering.

- Under träning så sätts vissa noder slumpmässigt till att vara ”döda”. D plockas temporärt bort ut nätverke så nätverket måste klara sig utan dem.
- Varje iteration så sätts en ny slumpmässigt utvald delmängd noder till att vara ”döda”.
- När vikten är bortplockad så har den ingen påverkan på outputen och vikten uppdateras inte heller vid backprop.



(a) Standard Neural Net

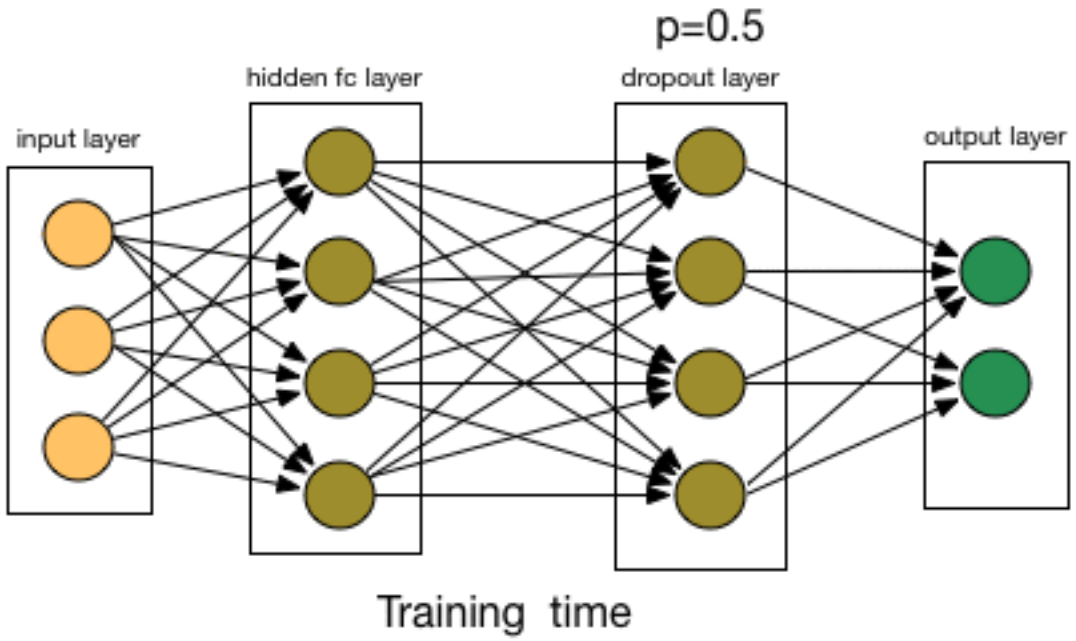


(b) After applying dropout.

Drop out

Drop out appliceras bara under träning –
under inferens så används hela nätverket utan drop out.

- Detta minskar beroendet mellan noder och nätverket blir mer robust och generaliserar bättre.
- Eftersom mindre information propageras genom nätverket när några noder är ”döda”, så tvingar det också nätverket att hitta fler oberoende representation av datan. Vi tvingar då noderna att hitta olika mönster i datan beroende på vilken information som de får se, trots att det är samma input/output. Detta leder till att nätverket blir effektivare och generaliserar bättre.



Data augmentering

- Att utöka sitt dataset, genom t.ex. data augmentering, kan också verka regulariserande.
- Generellt gäller: mer data* --> bättre generalisering.

* av god kvalité

Hyperparameteroptimering

- Hyperparametrar är parametrar som bestämmer över nätverket och tärningsprocessen på en hög nivå.
- Dessa lär sig nätverket INTE själv, utan sätts innan träning.
- Nätverksparametrar ≠ hyperparametrar

Exempel på hyperparametrar

- Learning rate
- Vilken optimeringsalgoritm som används (adam, rmsprop, sgd..)
- Antal träningsepoker
- Batch size (ofta bra att välja 2^x för bra utnyttjande av dator-resurser)
- Vilka aktiveringsfunktion(er) som används
- Dataset split ratio (train/validation/test)
- Antal gömda lager
- Drop-out rate
- Osv.....

Grid search

- Vi testkör modeller med massa olika hyperparametrar och sen testar vilken som blev bäst. Den bästa kombinationen av hyperparametrar är den vi använder vid den ”riktiga” träningen.

- Tänk på att det snabbt blir väldigt många körningar om man ska undersöka många kombinationer av hyparparametrar!