Nätverk inom AI: Gradient descent

729G83

LINKÖPINGS UNIVERSITET

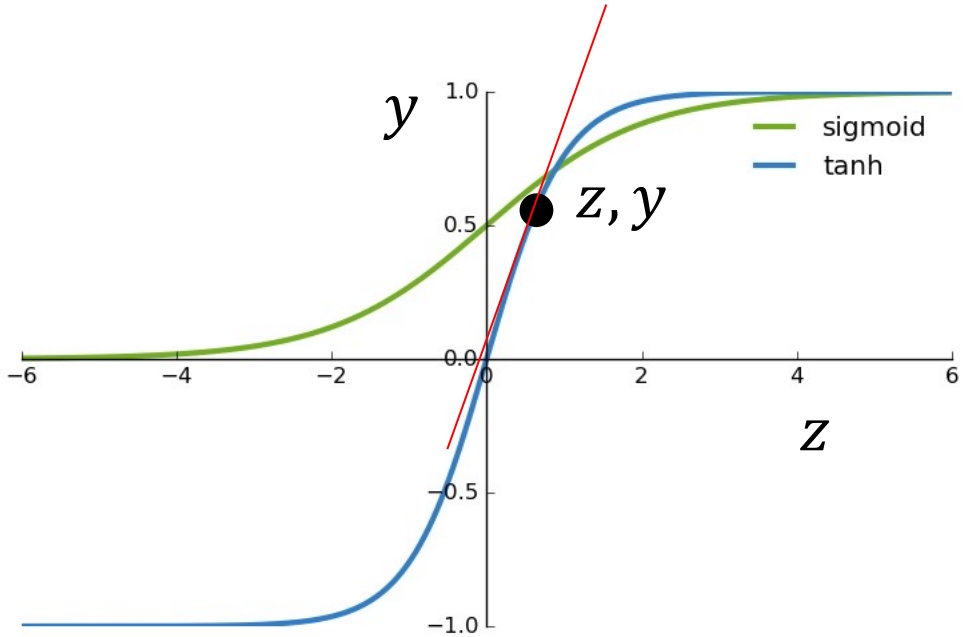
Översikt

- Gradient descent
  - Problem som kan uppstå
  - Lösningar
- Overfitting - underfitting
- Regularisering

Gradient descent

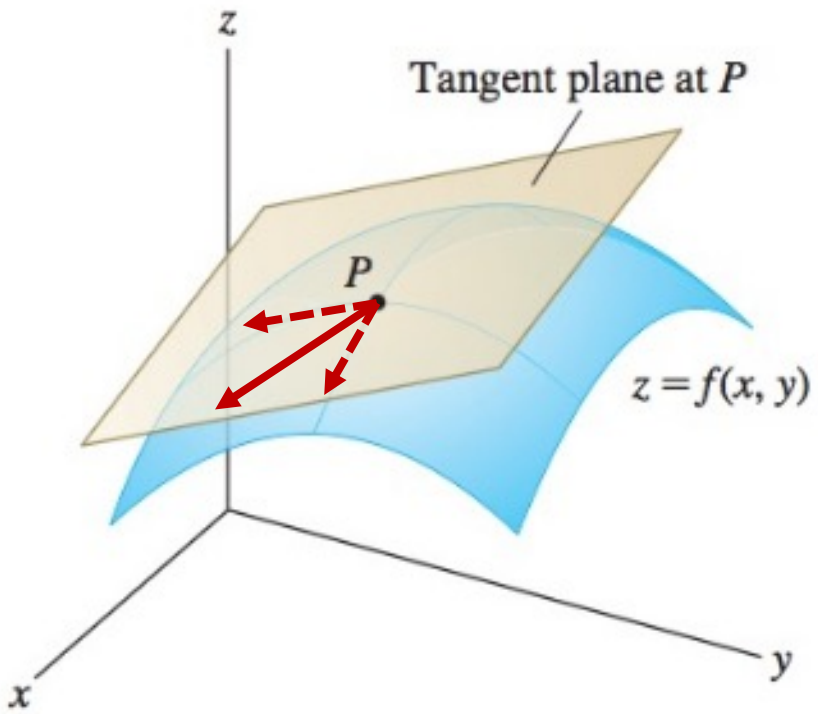
Tangent

- Om y ska minskas, hur ska z ändras?
  - Vill ha lutningen, i punkten z, y
- Derivatan av y m.a.p. $z = \frac{dy}{dz} = \frac{\partial y}{\partial z} = y'$

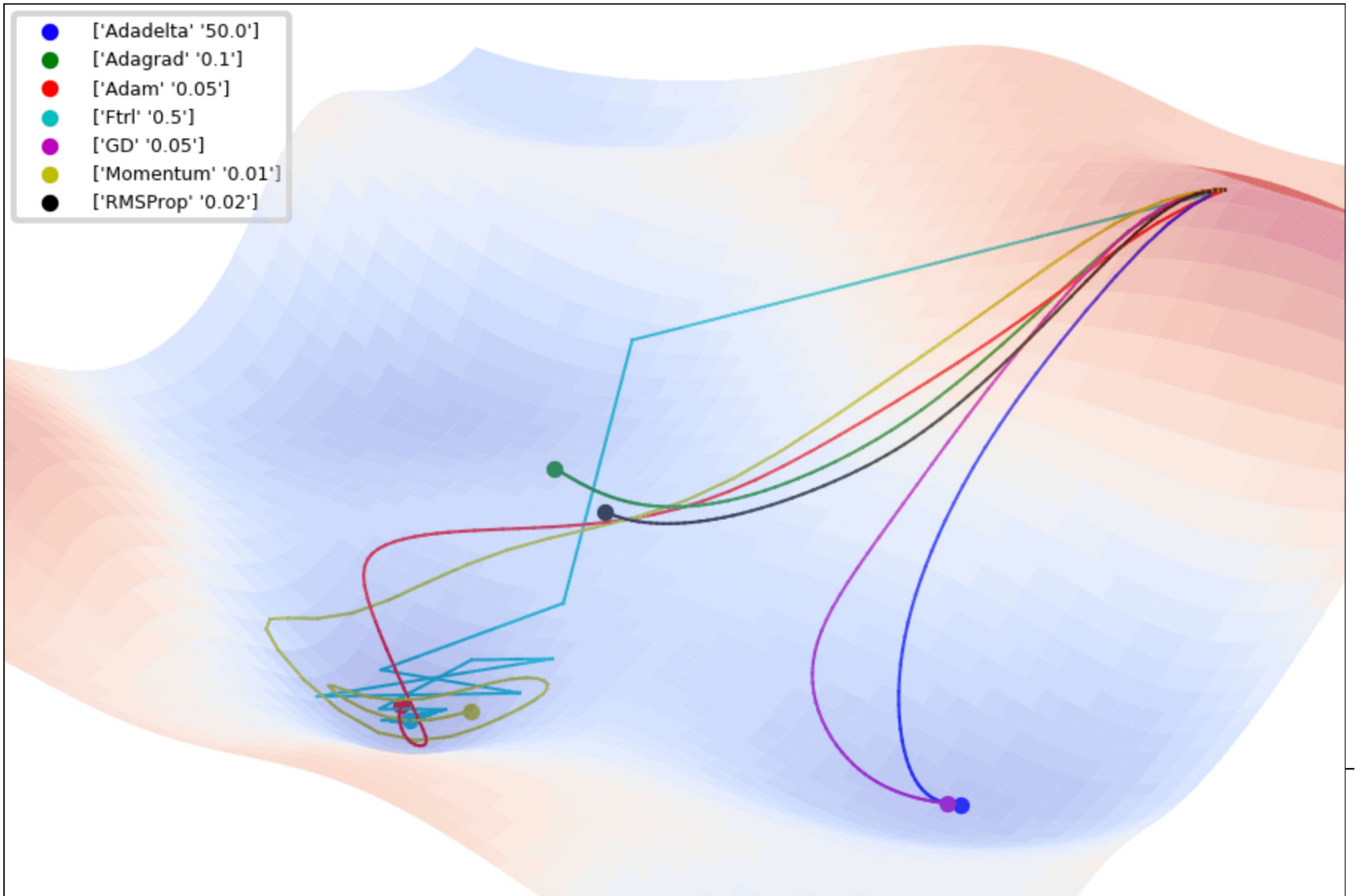


Loss är en flervariabelsfunktion

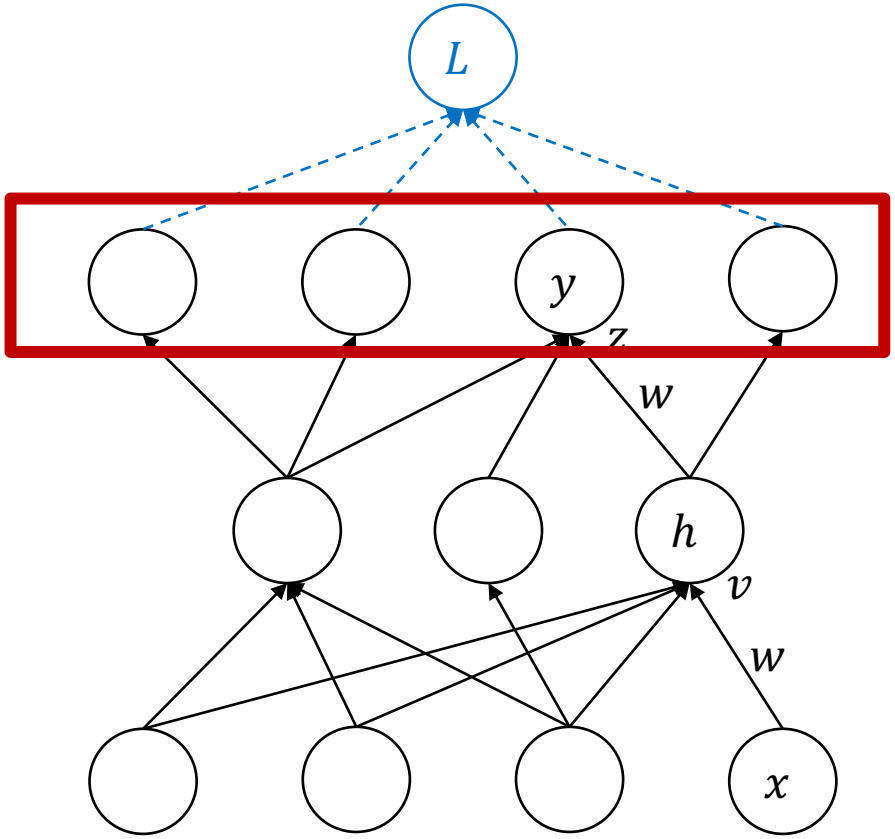
- Tangent (derivata) blir mångdim. yta



Visualisering av optimeringen



Bakåt-propagering



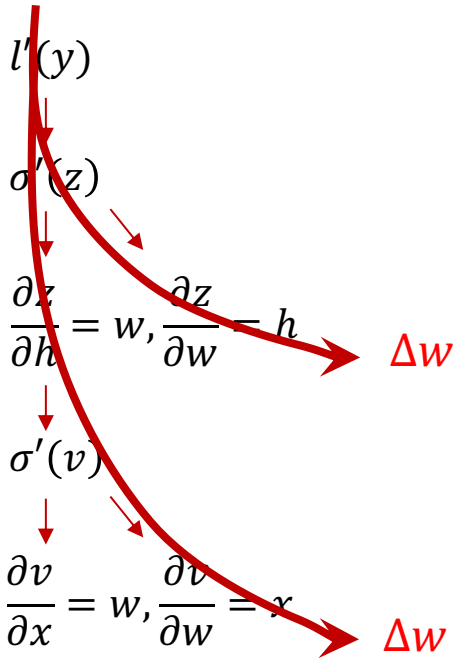
$L = l(y)$

$y = \sigma(z)$

$z = wh + b$

$h = \sigma(v)$

$v = wx + b$



$$\frac{\partial L}{\partial w} = l'(y)\,\sigma'(z)\,h\sigma'(v)x$$

$$\Delta w = \eta \frac{\partial L}{\partial w} = l'(y)\,\sigma'(z)\,w\sigma'(v)x$$

Exploding gradient

- Hög derivata som återkommer i flera lager
  - T.ex. i Recurrent Neural Networks (RNN) där samma derivata ”återanvänds” många gånger
  - Kommer att multipliceras många gånger om…
- Märks, t.ex. genom att loss har blivit NaN
- Märks om träningen åt pipan
  - Ryckig kurva som pekar uppåt
  - Jättestora viktuppdateringar efter varje mini-batch

Vanishing gradient

- Problem om något lager har låg derivata
  - För att vikterna är genomgående låga
  - För att aktiveringarna i ett lager har hamnat i platå-delen av aktiveringsfunktionen
- Märks på att inlärning avstannat (eller inte kommit igång)

Lösning

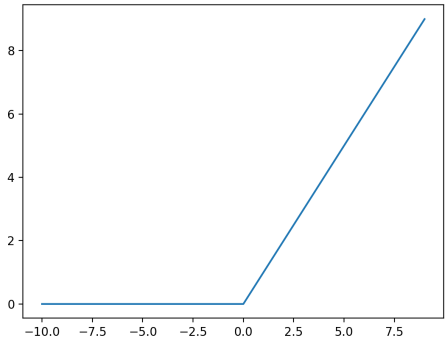
- Undvik platå-områdena på akt.funktionen
  - $netin = Wx + b$
  - Vill ha $netin$ inom ett bra intervall kring 0
  - Både aktiveringar och vikter behöver ligga inom bra intervall
    - I hela nätet!

Initialisering av vikter

- Vill inte ha vikter $= 0$ eller $= 1$
- Xavier-initialisering av vikter
  - Slumpvikter mellan 0 och 1
  - Dividera inkommande vikter med antal moder S i sändande lagret: W_{SR}/S^2
    - Många sändande noder → mindre vikter
    - Netin initialiseras inom stabilt intervall över olika lager
- För ReLU: dividera S med 2, så att $W_{SR}/(\frac{S}{2})^2 = 4W_{SR}/S^2$

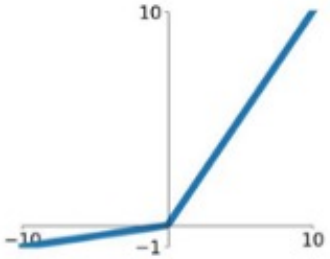
Undvik platå: ReLU el. Leaky ReLU

- ReLU $= max(0, x)$
- Derivatan $= 0$, när input ≤ 0
- Annars 1



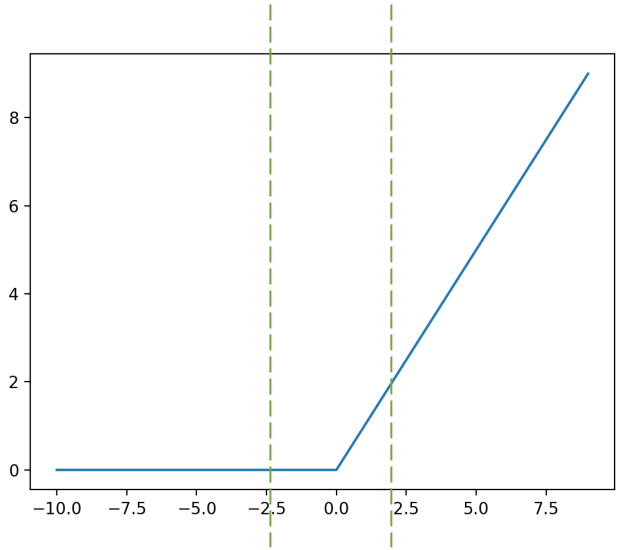
Leaky ReLU

$\max(0.1x, x)$



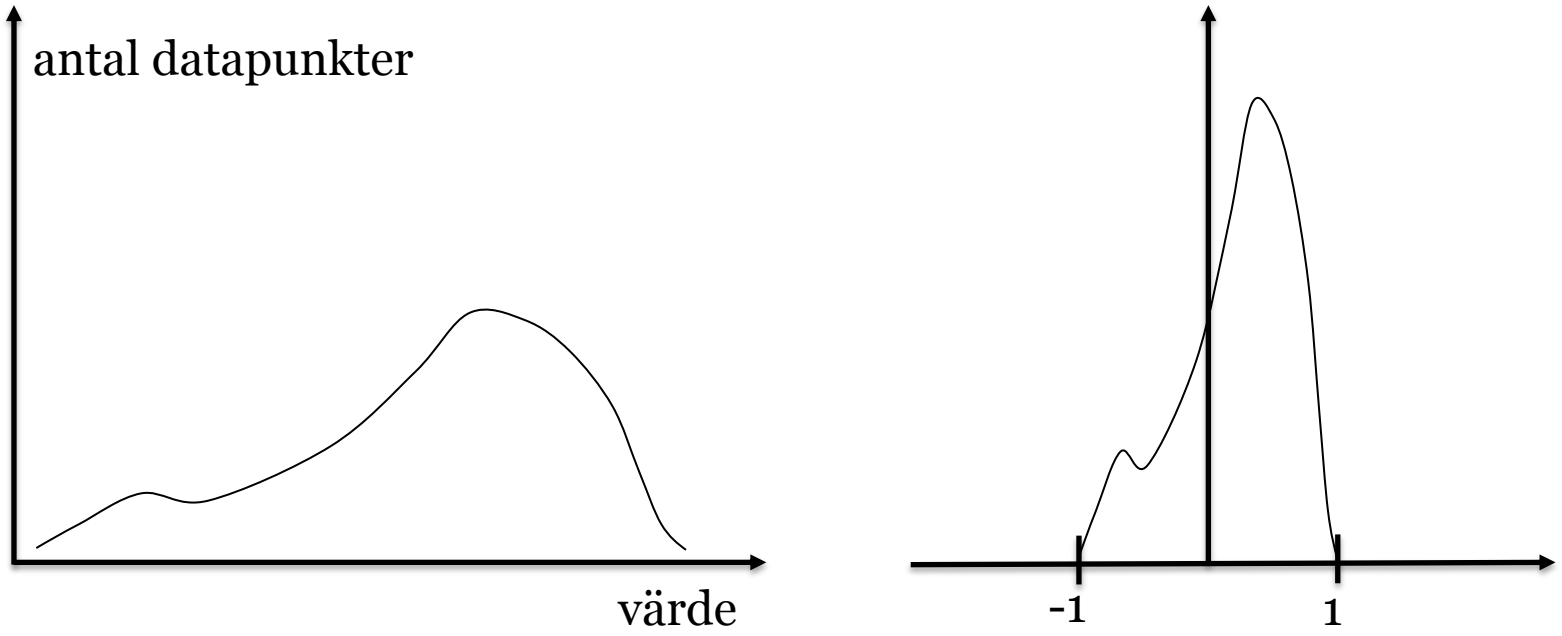
Vill hamna i rätt intervall

- Vill undvika platta delen av aktiveringsfunktionen
- Samtidigt viktigt att data träffar icke-linjära delen av aktiveringsfunktionen
  - Annars kollapsar nätet till ett enda linjärt lager



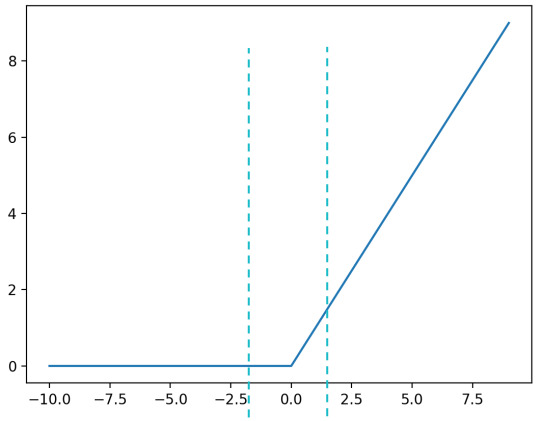
Normalisering av netin (och av indata)

- För indata, ofta (lite slappt)
  - Min-max scaling till [-1, 1]



Normalisering

- Vill få input inom rätt intervall
  - Centrerat kring 0
  - Varians (SD) = 1
  - *Zero-mean, unit variance*



Bra indatafördelning:
68% inom -1 och 1

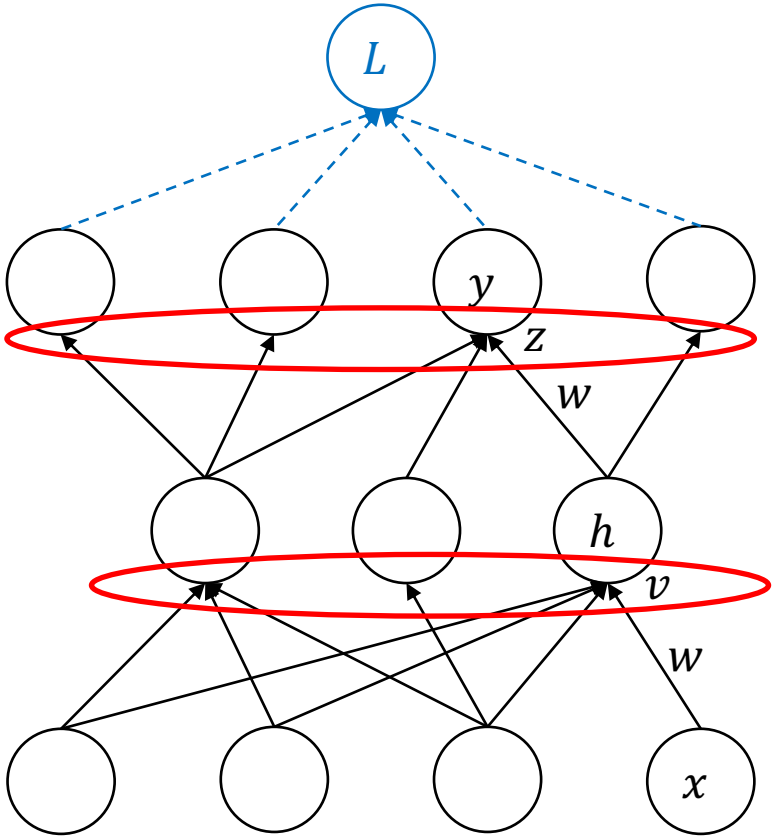
Normalisering

- För att centrera kring 0
  - Dra ifrån *snittet* från alla datapunkter
- För att få unit-variance
  - Dividera med *standardavvikelsen* för datamängden

Batch normalisering av (net) input

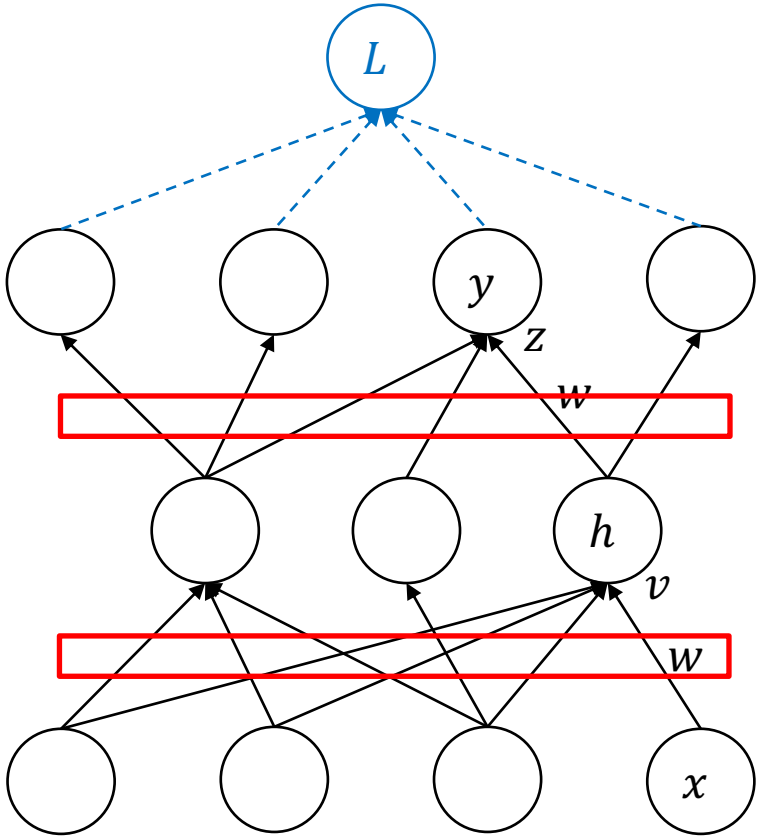
- Input = in till varje lager, dvs. även mellan lager
- Fördel: görs på basis av slumpvist urval av data
  - Statistisk fördelning varierar mellan batches
  - Stokastisk, dvs. slump-variation på input. Vi ”förvanskar” input på lite olika sätt för varje batch
    - Viss regulariseringseffekt
      - Mer robusta representationer
      - Mindre risk för överanpassning

Normalisering av netin för varje lager



Vill att netin ska vara inom rätt intervall

Normalisering av netin i Keras



Lägger BatchNorm före varje (dense) lager

Träning

Träning

- Använder ca. 70% av data
- Målet är att få bra test-resultat
  - Träningsresultat är *inte* intressant
  - (Det är underförstått att det har gått bra med träningen)

Träning

- Förutom vikter
- En uppsättning hyperparametrar som ställs in manuellt
  - T.ex. learning rate (och schedule)
  - Kolla om träningen fungerar
  - Dessutom, kolla om nätet uppför sig bra på valideringsdata…

Validering

- Valideringsdata = undanlagt från dataset:et
  - Typiskt 20 %
- Används *inte* för träning
- Används för att jämföra olika uppsättningar av hyper-parametrar
  - Brukar dessutom köra en valideringsepok efter varje träningsepok (för att bestämma när träningen ska stoppas)

Välj bra parametrar och bra modell

- Använd valideringsdata för att välja
  - Hyperparameterar, t.ex. lrate, lagerstorlek, etc.
  - Val.data används alltså för beslutsfattande
- Varför separata testdata?
  - ”Slöseri” på ytterligare 10% data…, eller?

Välj bra parametrar och bra modell

- Använd valideringsdata för att välja
  - Hyperparameterar, t.ex. lrate, lagerstorlek, etc.
  - Val.data används alltså för beslutsfattande
- Varför separata testdata?
  - Våra beslut och val har gynnat val.data!
    - Vi har optimerat mot val.data
  - Inte rättvisande att slut-testa med dessa data…

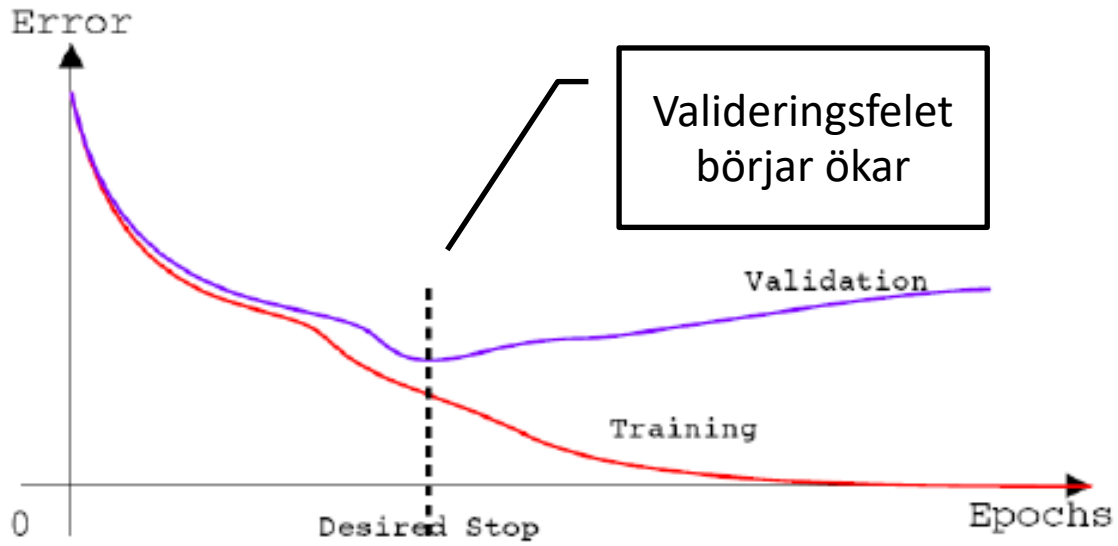
Test

- Testdata reserveras, t.ex. 10% av alla data (och körs *aldrig under träning*)
- Används *bara* när nätet har tränats och de bästa hyperparametervalen är gjorda
- Används bara en gång i slutet!
  - För att utvärdera nätets generaliseringsförmåga
- All annan användning medför risk att vi optimerar mot testdata

Hands-on praktik

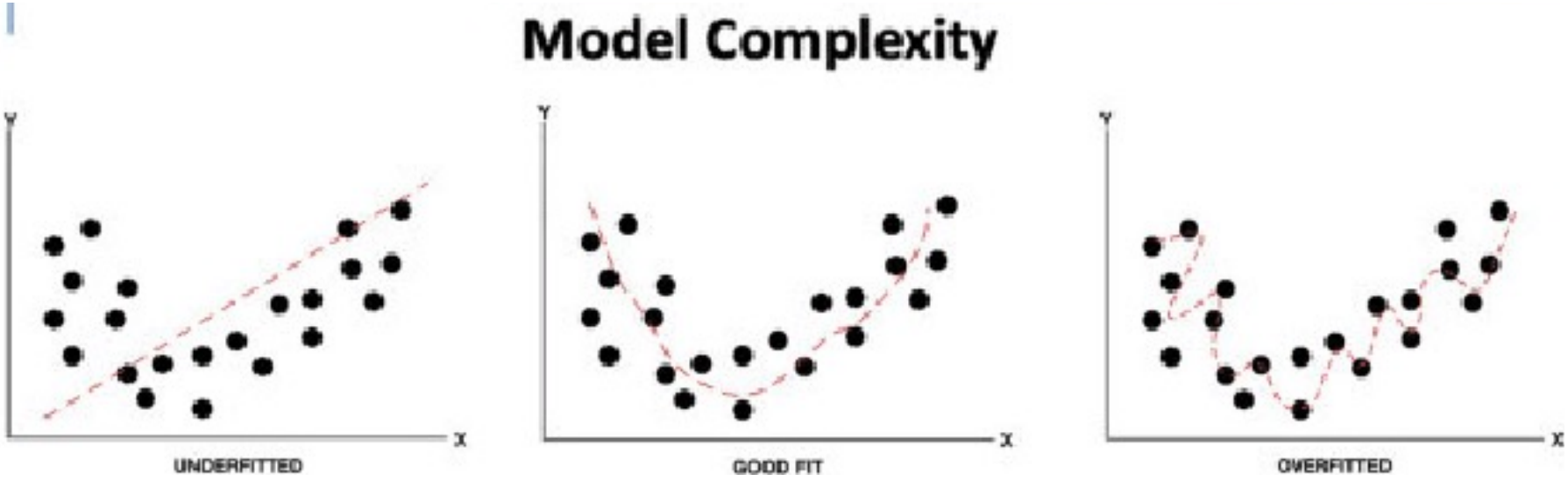
Best practice

- Testa med valideringsdata efter varje träningsepok
  - Epok = köra igenom alla data (samples) en gång
- Mål: stoppa träning om tecken på overfitting



Underfitting - overfitting

- Overfitting: Vill undvika att nätet har överkapacitet (och tillåts träningstid) att anpassa sig till varje knyck i data



Overfitting

- När nätet lär sig ovidkommande detaljer i data
- Överkapacitet i nätet i förhållande till datamängden
- För att undvika
  - Inte överträna
  - Bra förhållande mellan nätets komplexitet (antal lager, antal vikter) och tillgången till data
    - Små data → små nätverk!
  - Regularisera (dynamiskt begränsa nätverket)

Overfitting

- I stället för att skära ner i nätet för hand i förväg
  - Kan ha tagit bort för mycket
  - Tagit bort på fel ställen
- Regularisera
  - T.ex. öka loss när nätet använder sig av många eller överlag stora vikter

Regularisering (= uppstyrning, reglering)

- Tvinga nätet att titta på övergripande mönster
  - Inte bry sig om enstaka variationer i indata
    - T.ex. variationer i belysning på bilder
- Bestraffa ökad komplexitet i den modell som utvecklas under träning
  - T.ex. bestraffa många stora vikter
- Tvinga nätet att ta hänsyn till underrepresenterade data
  - Bestraffa extra när X produceras felaktigt, när väldigt många X i förhållande till Y i dataset:et

Olika sätt att regularisera (1)

- L2 el. L1-regularisering
  - Bestraffa många, stora vikter (som extra term i loss-funktionen)
  - L1: summan av absolut storlek på vikter (weight ”kill”)
    - Premierar glesa vikter
  - L2: summan av kvadrerad storlek på vikter (weight decay)
    - Premierar små vikter

Olika sätt att regularisera (2)

- Dropout
  - Selektivt stänga av slumpmässigt valda vikter
  - Tvingar fram robusthet, stabila representationer
- Batch normalisering av input
  - Ger stokastisk variation i data
    - Olika batch:es ”förvanskas” olika
  - Tvingar nätet att fokusera på övergripande mönster, inte detaljer i data

Olika sätt att regularisera (3)

- Droppa hela lager
  - Förutsätter att det finns förbikopplingar
- Data augmentation
  - Egentligen ett sätt att utöka datamängden
    - Nätets komplexitet < datamängd storlek

Obalans (bias) i data

Obalans (”bias”) i data

- Ont om viss typ av data
- T.ex. många med BMI kring 25, men knappt några med BMI > 40 i dataset:et
  - Nätet kan snabbt lära sig att säga 25 för alla data, och ändå få bra tränings- och valideringsresultat
    - Bara fel i de 2% av fallen som BMI var > 40

Obalans (”bias”) i data: lösning

- Tvinga nätet att ta hänsyn till underrepresenterade data
  - Bestraffa extra när små BMI:n produceras felaktigt som kategori-svar
- Minska ner (den större klassen), eller utöka med artificiella data (den mindre klassen)
  - T.ex. förskjuta bilden en aning så att objektet inte alltid är centrerat, rotera en aning, zooma, etc.
    - Ger många fler kattbilder för träning och test

Obalans (”bias”) i data: lösning

- Lägg till syntetiska data, t.ex. datorgenererade kattbilder
- Förträning på liknande dataset
- Om det är ont om labeled data
  - Förträning mha oövervakad (self-supervised) inlärning
  - T.ex. förutsäg nästa frame i en video
  - Sedan träna (transfer learning) på att kategorisera video

Obalans (”bias”) i data

- Annan statistisk fördelning av data mellan träning och test
  - T.ex. få katter i träningsdata, men väldigt många katter i testdata
  - Tvungen av praktiska skäl, t.ex. tränat självgående buss på parkeringsplatsen… ska köra ute på gatan
  - Börsen har gått upp sedan nätet tränades
- Lösning:
  - Monitorering av fel under drift
  - Kontinuerlig/periodisk omträning av nätet

} MLOps

rita.kovordanyi@liu.se

www.liu.se

LINKÖPINGS UNIVERSITET