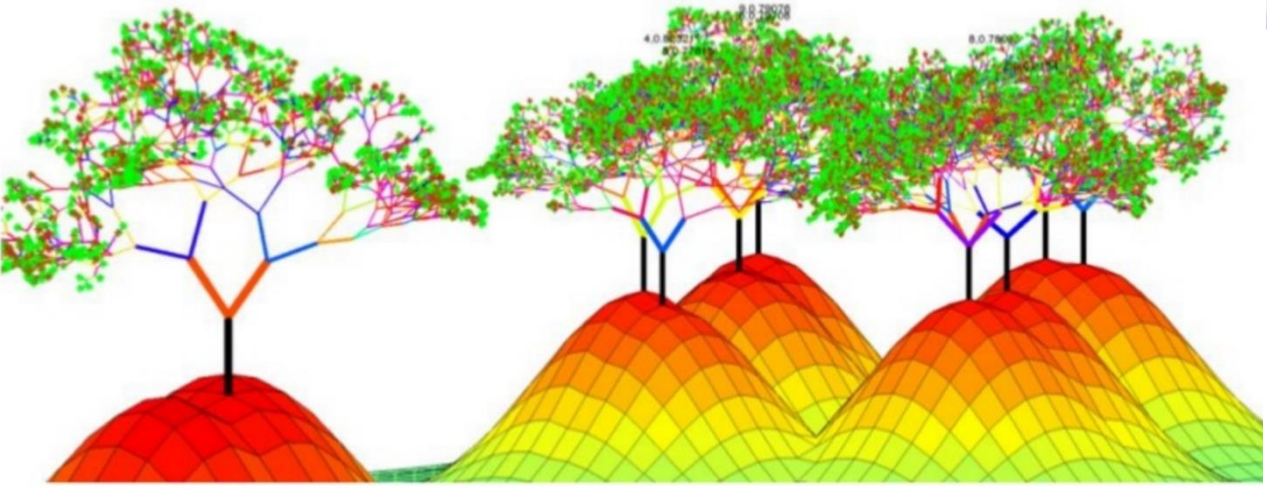
Norsk Regnesentral
NORWEGIAN COMPUTING CENTER

Del 3
- Beslutningstrær
- Random forest
- XGBoost

Webinar FFI

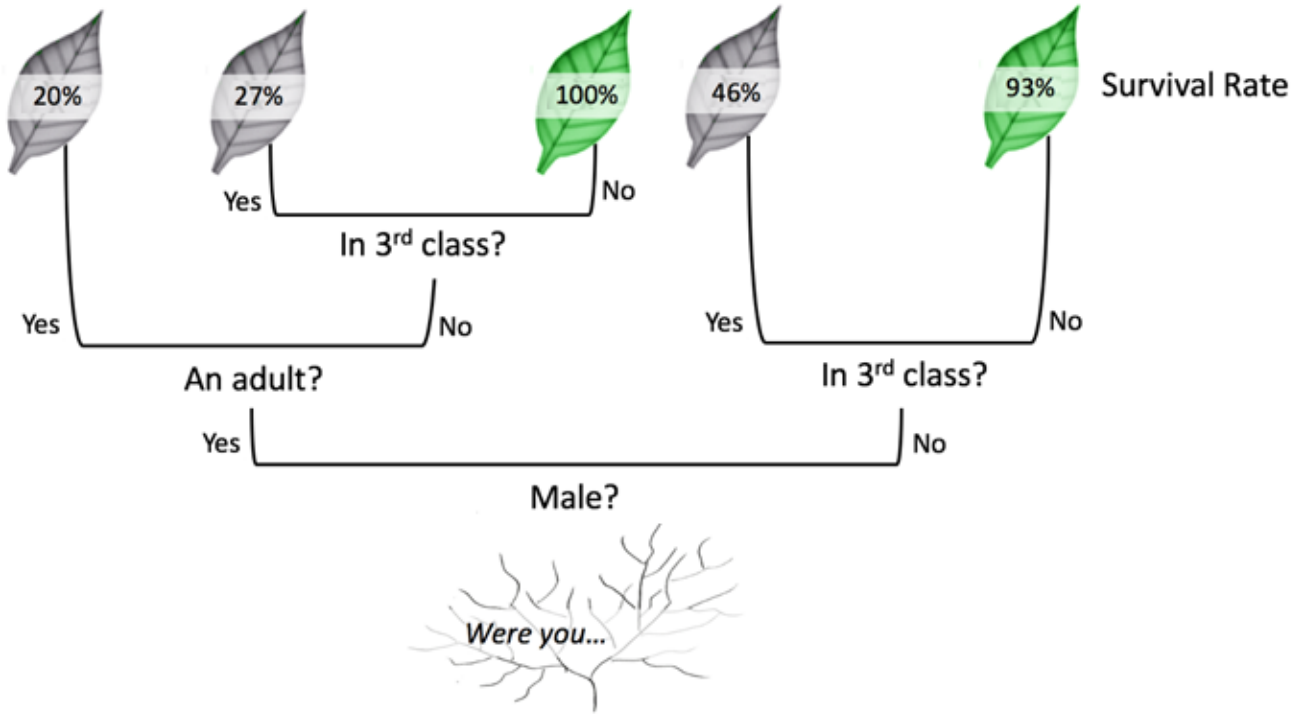
Introduksjon til sentrale metoder i statistisk modellering og maskinlæring

Martin Jullum
jullum@nr.no



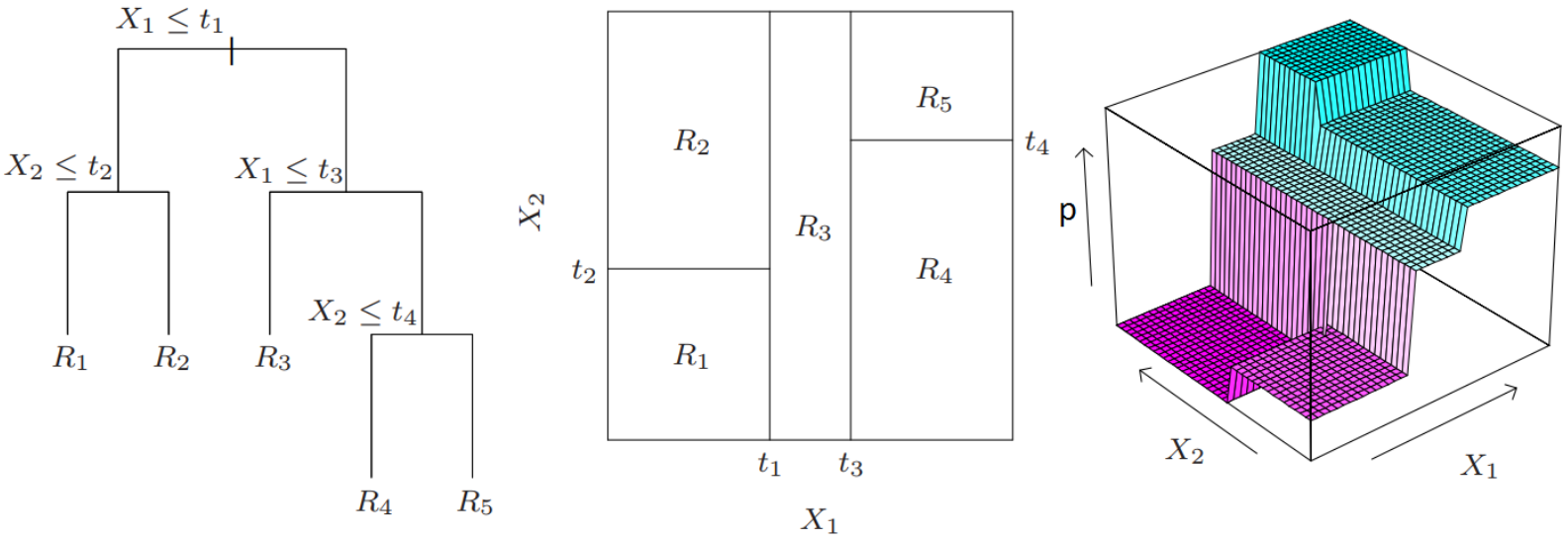
Beslutningstrær (I)

- Verdens enkleste nyttige statistiske modell!
  - Hver forgrening er basert på et JA/NEI-spørsmål for **<u>én variabel</u>**
  - Fungerer både for kontinuerlig og binær respons, samt klassifisering
  - Modellkompleksitet styres ved dybden av tree, antall blader,…



Trent tre-modell for overlevelsesrate på Titanic

Beslutningstrær (II)



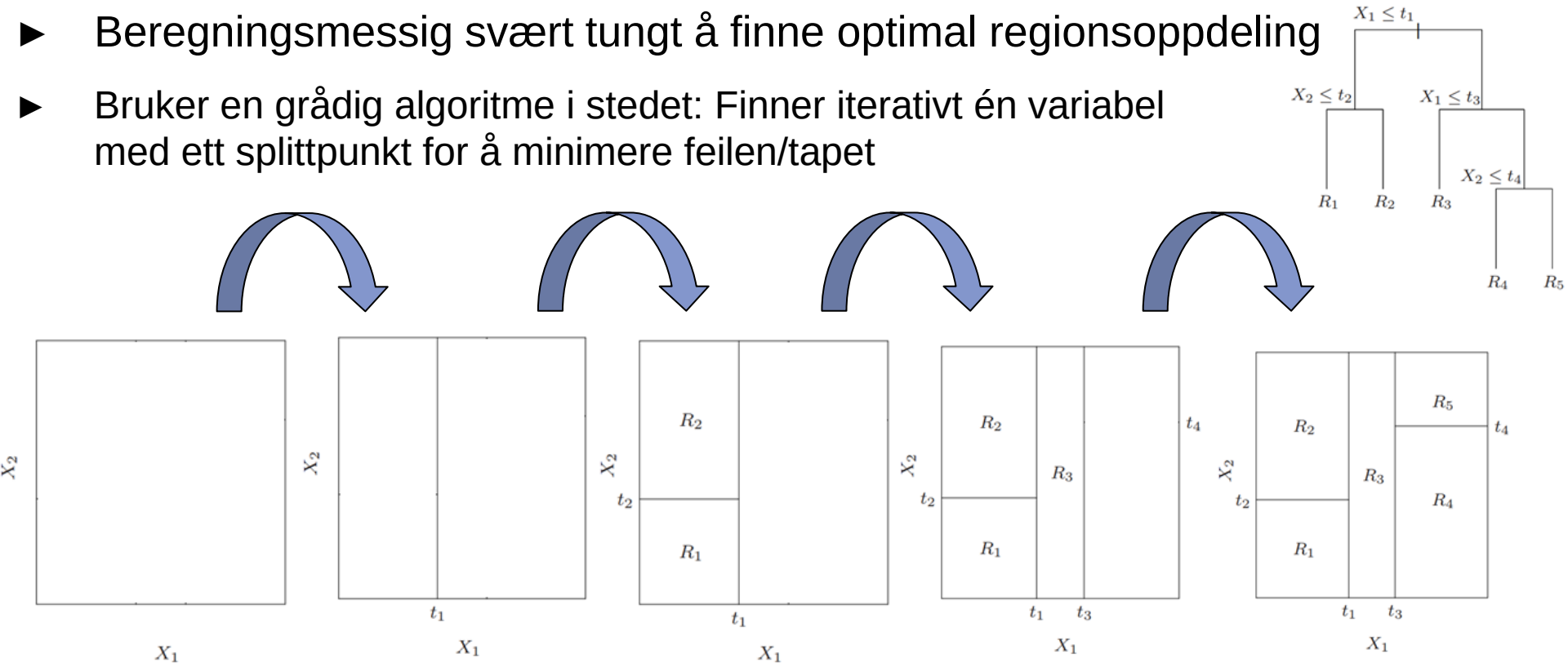
3 ulike visualiseringer av samme tre-modell

- Kan skrives som en vektet sum av indikatorvariable over regionene:

$$f(x) = \sum_{j=1}^{T} \theta_j \, 1_{\{x \in R_j\}}$$

Trening av tre-modeller

- Beregningsmessig svært tungt å finne optimal regionsoppdeling
- Bruker en grådig algoritme i stedet: Finner iterativt én variabel med ett splittpunkt for å minimere feilen/tapet



- Stopper basert på
  - (Kryss)validering
  - Bestemt dybde/antall blader
  - Når tapsreduksjonen er liten ved videre splitt

Egenskaper med tre-modeller

- Fordeler
  - Enkle å tolke
  - Enkle å trene
  - Invariant til monotone transformasjoner av variablene
  - Håndterer naturlig kontinuerlige og kategoriske data
  - Kan håndtere manglende data
  - Modellerer ikke-lineariteter og interaksjoner direkte
  - Skalerer godt til store datamengder
- Ulemper
  - Fort gjort å overtilpasse
  - Diskrete prediksjoner
  - Begrenset prediksjonskraft

Bagging

- Bagging = Bootstrap aggregating, Breiman (1994)
  - Modellblandingsteknikk som øker prediksjonskraften til enkeltmodell ved å ta et gjennomsnitt av mange enkeltmodeller tilpasset på *bootstrappede* trekk fra treningssettet

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)$$

- Bootstrapping

Trekke tilfeldige sett av observasjonene **med tilbakelegging**

Noen observasjoner blir kopiert opp, mens andre blir slettet

Data: 1 2 3 4 5

1. trekk: 3 2 5 3 4
2. trekk: 1 3 2 2 2
3. trekk: 5 3 5 2 2
4. trekk: 4 2 5 1 1
5. trekk: 2 1 5 1 3

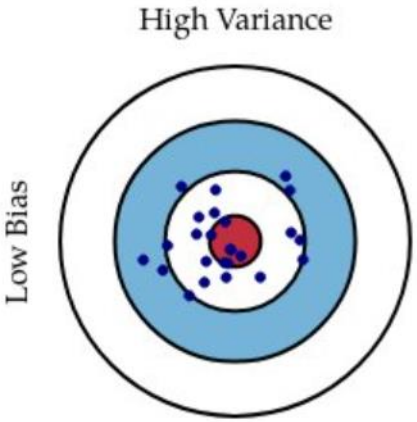
…

Random forest

- Bagging sikter på å redusere den totale variansen

$$Var(\frac{1}{2}(X+Y)) = \frac{1}{4}Var(X) + \frac{1}{4}Var(Y) + \frac{1}{2}Cov(X,Y)$$

- Bagging foretrekker modeller med lav bias og høy varians



- Random Forest, Breiman (2001)
  - Bagging med beslutningstrær
  - Ofte 100-1000 *dype* trær
  - Ekstra triks for å sikre ulike trær:
    - For hver splitt i hvert tre, trekk et tilfeldig utvalg av variabler som har lov til å være splittvariabel

Boosting: Prinsippet

- Modellblandingsteknikk som slår sammen mange enkle «basismodeller» $f_m(x), m = 1, \dots, M$ (weak learners) til en avansert (strong learner) $f_{final}(x)$
- Trener iterativt en og en basismodell, hver og en med mål om å reparere feilene til tidligere trente modeller (og minimere empirisk tap)
- Endelig prediksjon = Sum av prediksjoner fra alle basismodellene

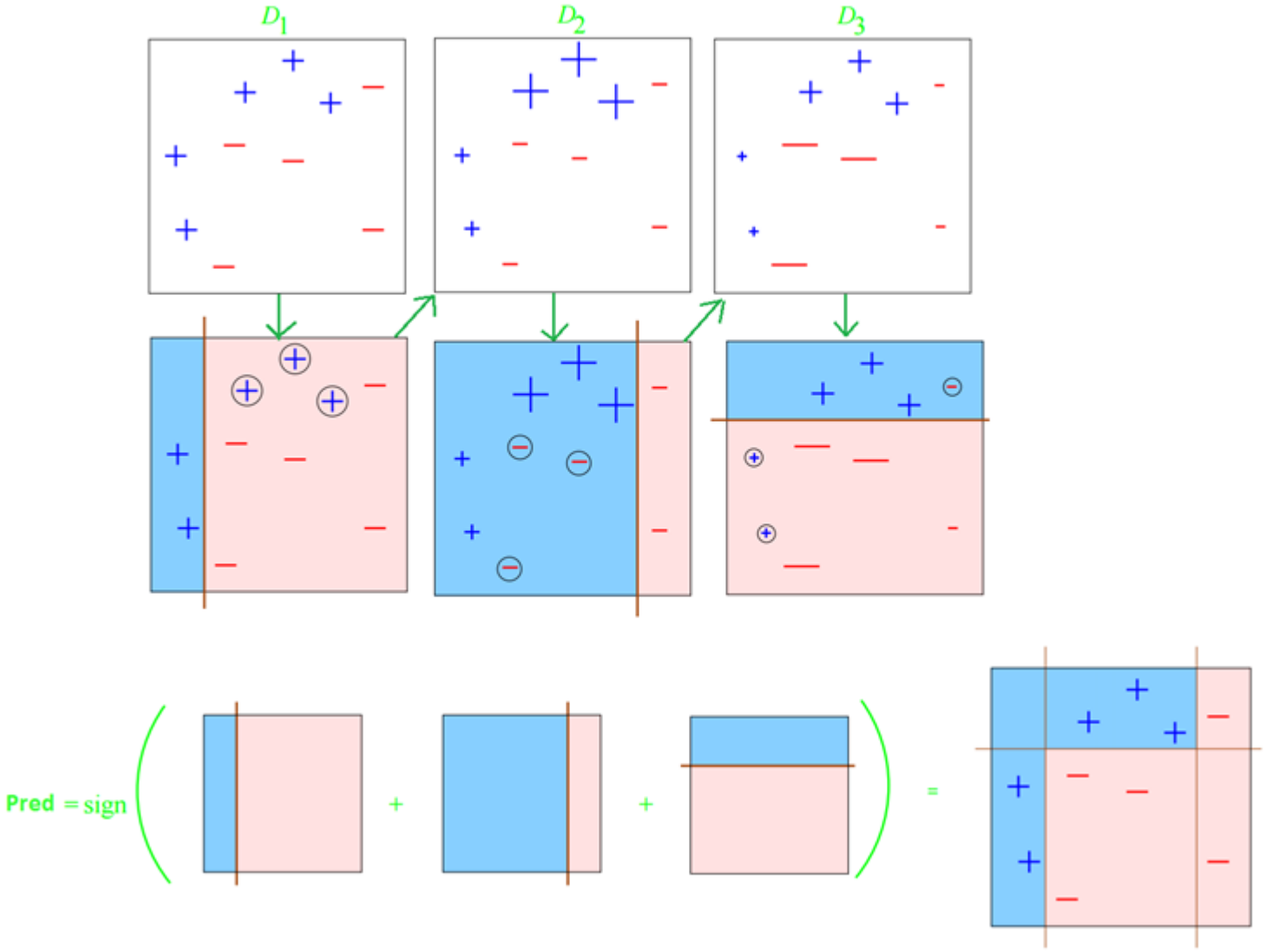
$$f_{final}(x) = f^{(M)}(x) = \sum_{m=1}^{M} f_m(x)$$

$$f_m = \underset{h \in \Phi}{argmin} \sum_{i=1}^{n} L(y_i, f^{(m-1)}(x_i) + h(x_i))$$

For modellklasse Φ og tapsfunksjon $L(\mathrm{x}, \mathrm{y})$

Typiske tapsfunksjoner
Regresjon: $L(\mathrm{y}, p) = (\mathrm{y} - \mathrm{p})^2$
Binær klassifisering: $L(\mathrm{y}, p) = y \log(p) + (1 - y)\log(1 - p)$

Eksempel boosting



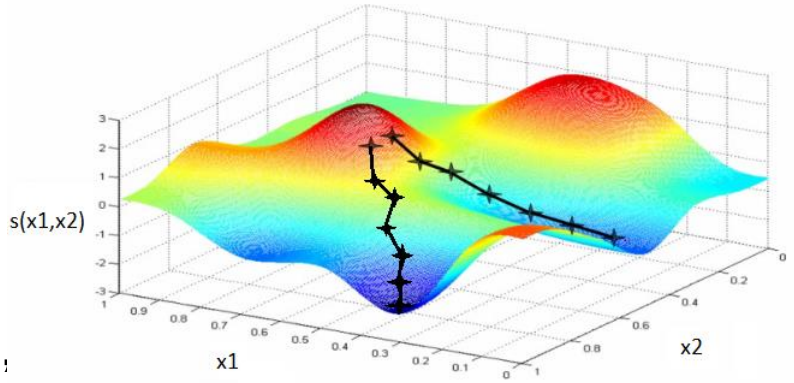
Egenskaper med boosting

- Fordeler
  - Arver typisk alle egenskapene til basismodellene, men gir en vilkårlig god prediksjonskraft i tillegg
- Utfordringer
  - Svært viktig å kontrollere overtilpasning for å få en god modell
  - Boosting kan i seg selv ikke parallelliseres
  - Generelt vanskelig å oppdatere med nye modeller via

$$f_m = \underset{h \in \Phi}{argmin} \sum_{i=1}^{n} L(y_i, f^{(m-1)}(x_i) + h(x_i))$$

Gradient boosting (machine)

- Gradient descent
  - Iterativ metode for å finne minimum av multivariat funksjon $s(x)$
  - Tar steg langs den negative gradienten: $x_m = x_{m-1} - \rho_m s'(x_{m-1})$



- Gradient boosting = Gradient descent for funksjoner/modeller
- Vi vil minimere $$f_m = \underset{h \in \Phi}{argmin} \sum_{i=1}^{n} L(y_i, f^{(m-1)}(x_i) + h(x_i))$$
- La $s_i(z) = \mathrm{L}(y_i, z), i = 1, \ldots, n$
- Bruk gradient descent på hver s_i
- Finn nærmeste modell ved å minimere

$$\underset{\rho, h \in \Phi}{arg\,min} \sum_{i=1}^{n} \left(s_i' \left(f^{(m-1)}(x_i) \right) - \rho h(x_i) \right)^2$$

- Gradient boosting machine = Gradient boosting med tre-modeller

Bagging vs boosting med tre-modeller

- Bagging
  - Sikter mot å redusere total varians
  - Foretrekker modeller med lav bias (+ høy varians)
  - Trener uavhengige modeller – enkelt å parallellisere
- Boosting
  - Sikter mot å redusere total bias (weak learner -> strong learner)
  - Foretrekker modeller med lav varians (+ high bias)
  - Trener avhengige modeller – sekvensielt
- Dype (bagging) and korte (boosting) trær er godt egnet pga dere fordelaktige egenskaper
  - Ulempene med beslutningstrær reduseres når mange kombineres

XGBoost = eXtreme Gradient Boosting

- Et open souce bibliotek bygget rundt en effektiv implementering av gradient boosting med tre-modeller som basismodeller
  - Utviklet av Tianqi Chen (Uni. Washington) i 2014
- Implementasjon
  - Grensesnitt for mange språk/plattformer: C++, Python, R, Julia, Java, Apache Spark etc.
  - Parallelliserbar trening av trærne, minnegjerrig og skalerbar
  - Kjører både på CPU og GPU
- Metodiske nyvinninger
  - 2.ordens approksimasjon av tapsfunksjonen – mer presis/effektiv enn ordinær gradient boosting
  - Legger til regularisering på toppen av original tapsfunksjon
- Praktisk bruk
  - Veldig mange parametere som *kan* skrus på, må gjøres manuelt
  - Kan ta lang tid å optimalisere/tune, men brukbare defaultparametere
  - «The Kaggle game killer»

Funksjonalitet i XGBoost

- Håndterer både kryssvalidering og ferdigoppdelt trening/validering/testsett
- Kan definere egne tapsfunksjoner og valideringsmål (mange allerede implementert).
- Kan følge valideringsresultater mens modellen kjører (f.eks. AUC på trening, validering og testsett)
- «Early stopping» (stopper å legge til nye trær når valideringsresultater ikke forbedres lenger)
- Mange tilgjengelige måter å håndtere overtilpasning på
- Ingen pre-prosessering/skalering/standardisering nødvendig
- Håndterer manglende data automatisk (lærer default retning i hver splitt)
- Effektiviserer trening av trær ved å forhåndsdefinere en begrenset mengde splittpunkter (histogram-metoden)

XGBoost – diverse

- Konkurrenter
  - LightGBM (Microsoft)
    - Har drevet/motivert mye av utviklingen av XGBoost
    - Mye likt, men ikke like modent og mangler noe funksjonalitet
    - Fortsatt noe raskere enn XGBoost?
  - CatBoost (Yandex)
    - Lignende, men håndtere også kategoriske variable direkte
    - Var langt treigere, men har blitt vesentlig bedre
    - Begrenset dokumentasjon
- Jeg har enda til gode å se et eksempel der Random Forest gjør det bedre enn en tunet XGBoost model!
- Hovedutfordringer:
  - Vanskelig/tidkrevende å finne optimal modell
  - Takler kun numerisk input: Ikke så god når det er mange kategoriske variable med mange klasser.

Ressurser

- Didrik Nielsen, Masteroppgave NTNU, 2016: https://brage.bibsys.no/xmlui/handle/11250/2433761
- Chen & Guestrin (2016), XGBoost: A Scalable Tree Boosting System: https://arxiv.org/abs/1603.02754
- Hastie et al. (2009), Elements of Statistical Learning, Ch 9.2 + 10
- XGBoost GitHub: https://github.com/dmlc/xgboost
- XGBoost dokumentasjon: http://xgboost.readthedocs.io
- Slides fra foredrag med Tianqi Chen: http://datascience.la/xgboost-workshop-and-meetup-talk-with-tianqi-chen/