CORTEVA™
agriscience

Technical nuances of machine learning:
Implementation and validation of supervised methods for genomic prediction in plant breeding

Alencar Xavier
Research Scientist at Corteva Biostatistics
Adjunct professor at Purdue University

http://dx.doi.org/10.1590/1984-70332021v21Sa15

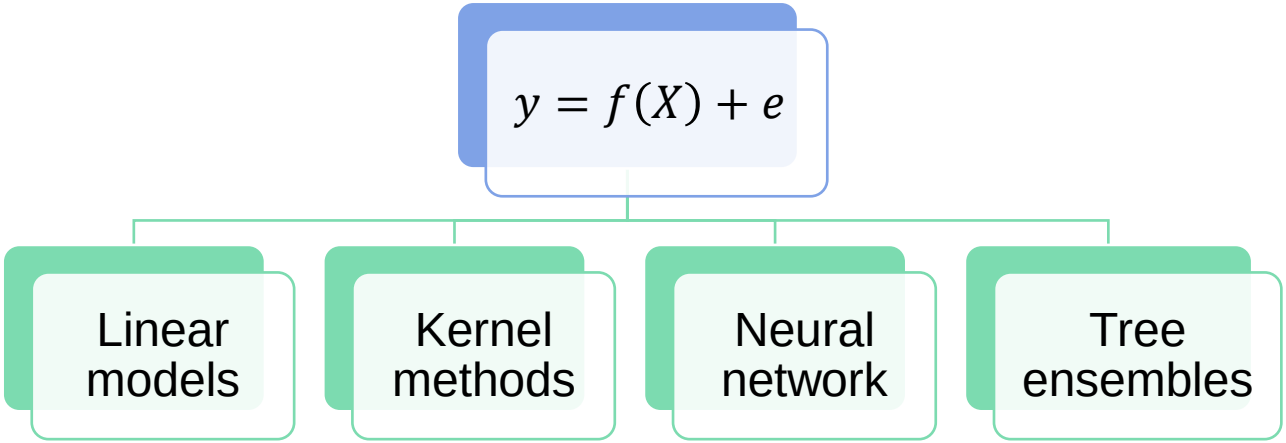
Outline

1. **Introduction**
2. **Machines**
   - Linear models
   - Kernel methods
   - Neural networks
   - Tree ensembles
3. **Validation**
   - Schemes and metrics
   - Information and case of study
4. **Conclusion**

1. **Introduction**
2. **Machines**
   - Linear models
   - Kernel methods
   - Neural networks
   - Tree ensembles
3. **Validation**
   - Schemes and metrics
   - Information and case of study
4. **Conclusion**

Introduction

Genomic prediction

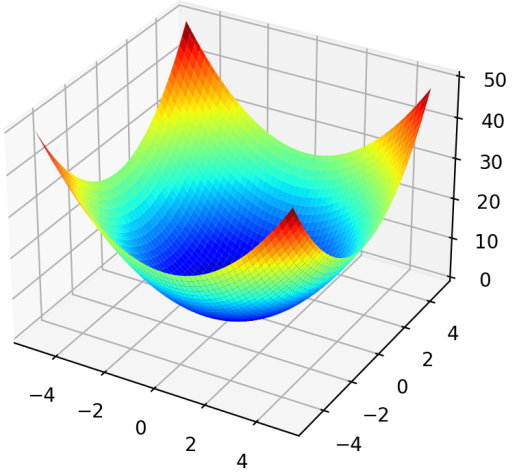


Objective of this presentation

- Describe machine learning methods without (too many) jargons
- Review validations strategies to contrast methods

Divergency in philosophy

- In quantitative genetics:
  - Parameters: Variance components + Regression coefficients
  - Function: Likelihood (complex and convex)
  - Tuning: Generally, not needed
  - **Method**: First order (EM, MCMC), second order (AI, MIVQUE)
- In machine learning:
  - Parameters: Regression coefficients (NO VARIANCES!)
  - Function: MSE, L2 (simple)
  - Tuning: Cross validations, need secondary objective function
  - **Method:** First order: coordinate & gradient descent



1. Introduction
2. **Machines**
   - Linear models
   - Kernel methods
   - Neural networks
   - Tree ensembles
3. Validation
   - Schemes and metrics
   - Information and case of study
4. Conclusion

Machines

1. Linear methods

$$y = X\beta + e$$

- Phenotype is described as a linear combination of markers
- Easy to compute; easy to store (vector β); easy to interpret
- LMs do not capture any patter that is not explicitly declared in X

Solution for linear models

Conditioning to univariate:
(Coordinate descent)

$$y = Xb + e$$
$$y = X_{-j}b_{-j} + x_j b_j + e$$
$$y - X_{-j}b_{-j} = x_j b_j + e$$
$$y_j = x_j b_j + e,$$

Univariate solutions for b_j

- $b_j(OLS) = \frac{x_j' y_j}{x_j' x_j}$ * Unique solution (1722)
- $b_j(RR) = \frac{x_j' y_j}{x_j' x_j + \lambda}$ * Unique solution (1970)
- $b_j(EN) = \frac{x_j' y_j - \lambda}{x_j' x_j + \lambda}$ (2005)
- $b_j(LAR) = \frac{MED(y_j \# x_j)}{Var(x_j)}$ (1935)
- $b_j(LASSO)_+ = \frac{x_j' y_j - \lambda}{x_j' x_j}$ (1996)

“Translation” table for geneticists

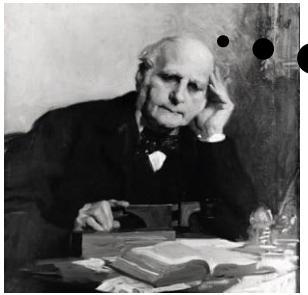
- Penalization = Shrinkage
- Multiple “penalizations” = Mixed model
- Least squares = Fixed effect
- Ridge regression = Random effect

The shrinkage parameter λ

- Quantitative genetics
  - Analytical solution: $\lambda = \sigma_e^2 \div \sigma_\beta^2$ (only applies to ridge)
  - Variances found via REML, Bayesian, others (MIVQUE, Tilde-Hat)
- Machine learning
  - Run a cross-validation to find λ
  - Secondary criteria to define the best λ… usually some metric of prediction

Solving: $\mathbf{y = Xb + e}$

$$Finding \rightarrow argmin(e'e + \lambda b'b)$$



I've created a monster!!

- Coordinate descent

  (Use diagonals of LHS)

$$\hat{b}_j^{t+1} = \frac{x_j'(y - X_{-j}\hat{b}_{-j})}{x_j'x_j + \lambda}$$

Used for WGR (RR, BayesA)

glmnet, BGLR, bWGR, GS3

- Gradient descent

  (Does not build LHS)

$$\hat{b}^{t+1} = b^t - \frac{2r}{n}\left[X'(y - X\hat{b}^t) + \lambda\hat{b}^t\right]$$

Used for Deep Neural Nets

TF/Keras, PyTorch, MXNet, h2o

- Second order

  (Builds entire LHS)

$$\hat{b} = (X'X + \lambda)^{-1}(X'y)$$

Used for everything else

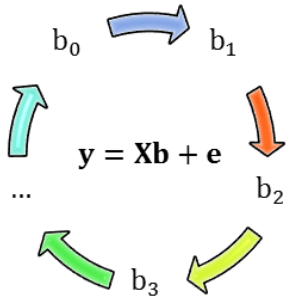
ASREML, lme4, SAS, BLUPF90

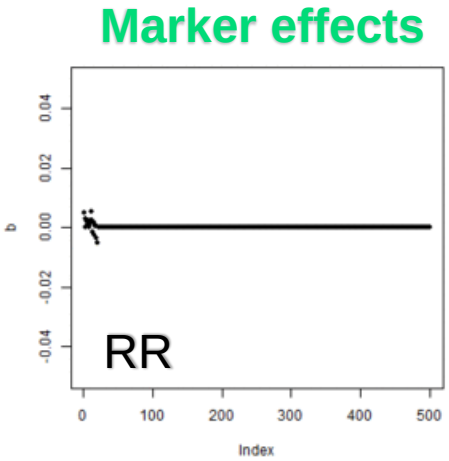
Coordinate descent of ***ridge regression*** (RR) and ***elastic net*** (EN)

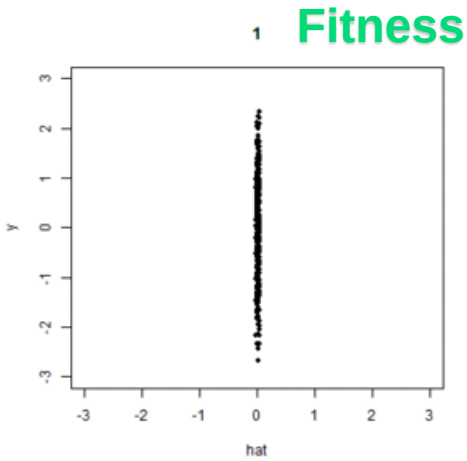
CD solved using Gauss-Seidel:

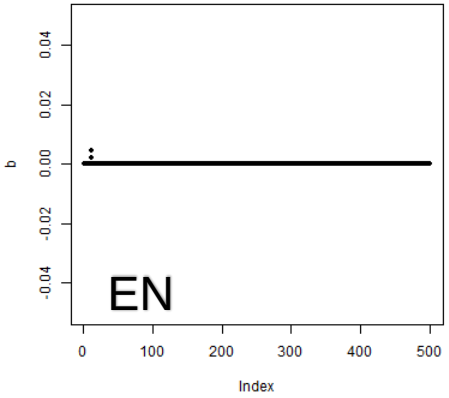
$$\hat{b}_j^{t+1} = \frac{x_j'\hat{e}^t + x_j'x_j\hat{b}_j^t}{x_j'x_j + \lambda}$$

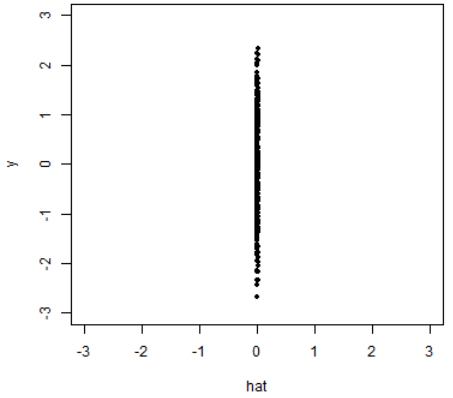
$$\hat{e}^{t+1} = \hat{e}^t - x_j\left(\hat{b}_j^{t+1} - \hat{b}_j^t\right)$$





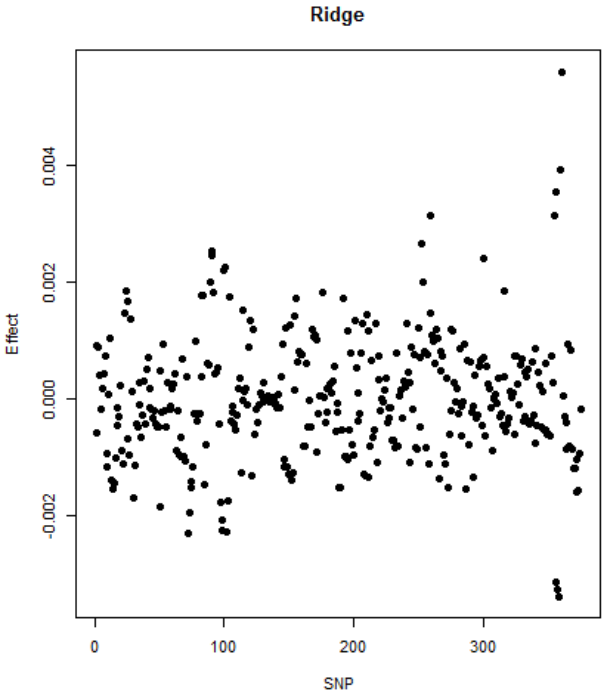


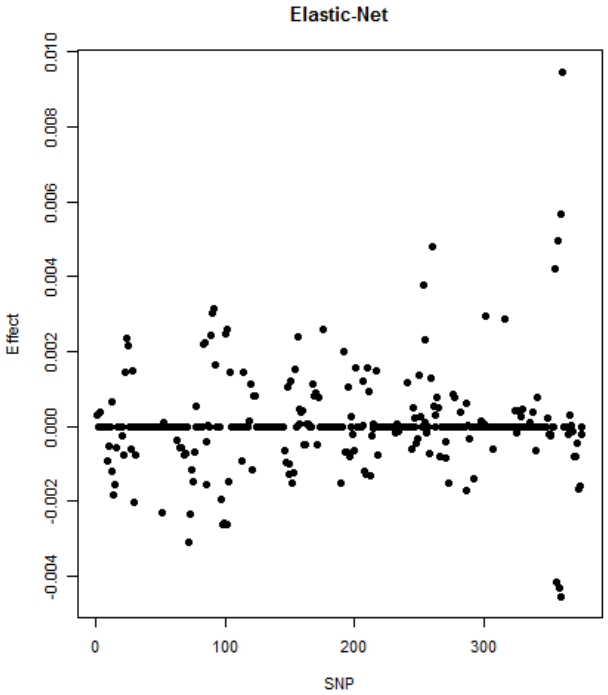


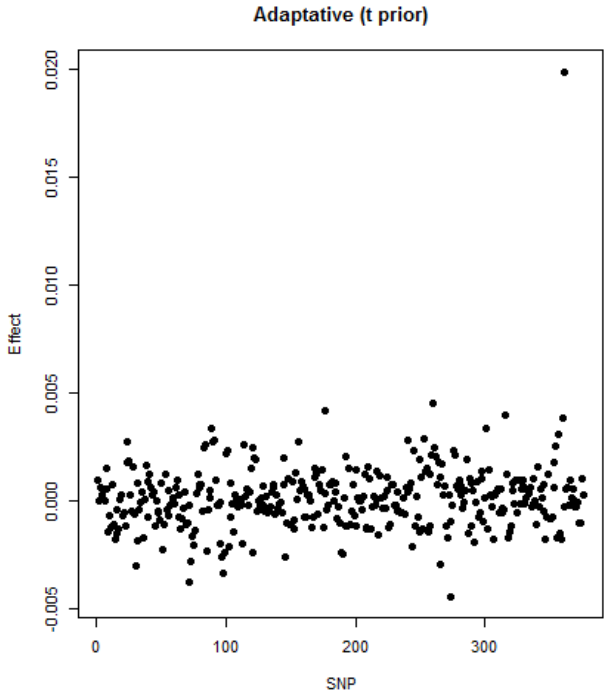


CORTEVA agriscience
Alencar.Xavier@Corteva.com
Corteva Biostatistics, Methods group

Impact of different models on the marker effects







Dataset ***tpod*** from bWGR has one QTL on Chr19

Alencar.Xavier@Corteva.com
Corteva Biostatistics, Methods group

2. Kernel methods

$$y = K(X) + e$$

- Markers are used to “map” individuals by similarity / relationship
- Capture complex patterns; Good for P>N and interactions; Solved as LMs;
- Requires factorization; Not possible to store the model from K (“lazy learner”)

Creating kernels

- Kernels (K) are transformations of X
- Before solving the model, we must compute $K = f(X)$

Kernel	Linear	Linear interaction	Gaussian (RBF)	Arccosine
$f(X)$	$K = \alpha XX'$	$K = \alpha(XX')\#(XX')$	$K = \exp(-\alpha D^*)$	$K(i,j) = \pi(x_i'x_i)(x_j'x_j)\sin(\theta_{ij}) + (\pi - \theta_{ij})\cos(\theta_{ij})$
Parameter	$\alpha = n^{-1}tr(XX')$	$\alpha = n^{-1}tr[(XX')\#(XX')]$	$\alpha = median(D)$ Tuning?	$\theta_{ij} = cor^{-1}Corr(x_i, x_j)$
Known as	GBLUP, Kalman filter	Epistatic kernels	RKHS, SVR	“Deep kernel”

* D = Euclidean distance = $\sum(x_i - x_j)^2$

Solution for kernel methods

	None	Inversion	Eigen (Spectral)	Cholesky
	$y = g + e$ $y \sim N(0, K\sigma_g^2 + I\sigma_e^2)$	$y = g + e$ $y \sim N(0, K\sigma_g^2 + I\sigma_e^2)$	$y = Ua + e$ $y \sim N(0, UDU'\sigma_g^2 + I\sigma_e^2)$	$y = La + e$ $y \sim N(0, LL'\sigma_g^2 + I\sigma_e^2)$
Factorization	$f(K) = K$	$f(K) = K^{-1}$	$f(K) = UDU'$	$f(K) = LL'$
Solution	$(K + \lambda I)g = Ky$	$(I + \lambda K^{-1})g = y$	$(I + \lambda D^{-1})a = U'y$	$(L'L + \lambda I)a = L'y$
Genomic prediction	$g = g$	$g = g$	$g = Ua$	$g = La$

When K is too big to factorize

K^{-1} sometimes can be estimated directly from data (e.g., pedigree)

1) When K is not inversible;
2) Speed up convergence;
3) Reduction of dimensionality;

Cheaper than Eigen, but K must be inversible

Prediction of new individuals?

- Approach 1 - **Conditional expectation**
  - $g_{new} = K_{obs,new} K_{obs}^{-1} g_{obs}$
  - $g_{new} = K_{obs,new} \sigma_g^2 \left(K_{obs} \sigma_g^2 + I \sigma_e^2 \right)^{-1} y_{obs}$

- Approach 2 - **Missing value**
  - Fit the model where K has both observed and new individuals

3. Neural networks

$$y = \alpha(\alpha(XB_1)B_2)b_3 + e$$

- Markers are used to create non-linear latent spaces
- Can capture complex patterns; Deal with large datasets;
- Requires extensive tuning; Not objectively interpretable;

Progression from LM to Deep Neural Network

Models illustrated without intercept

Linear model

$$y = Xb + e$$

PLS/PCR model

$$y = (XB_1)b_2 + e$$

NN model

$$y = \alpha(XB_1)b_2 + e$$

Deep NN model

$$y = \alpha(\alpha(XB_1)B_2)b_3 + e$$

α = activation function

Alencar.Xavier@Corteva.com
Corteva Biostatistics, Methods group

Solution for neural networks

Let's start with Gradient descent for a simple ridge regression

$$y = Xb + e$$

$$\nabla = \frac{\partial (y - Xb)'(y - Xb) + \lambda b'b}{\partial b}$$

$$\hat{b}^{t+1} = \hat{b}^{t} - r\nabla$$

$$\hat{b}^{t+1} = \hat{b}^{t} - r\left[-2n^{-1}X'\left(y - X\hat{b}^{t}\right) + 2n^{-1}\lambda\hat{b}^{t}\right]$$

$$\hat{b}^{t+1} = \hat{b}^{t} - r\left(-2n^{-1}X'\hat{e} + 2n^{-1}\lambda\hat{b}^{t}\right)$$

$$\hat{b}^{t+1} = \hat{b}^{t} + 2rn^{-1}X'\hat{e} - 2n^{-1}\lambda\hat{b}^{t}$$

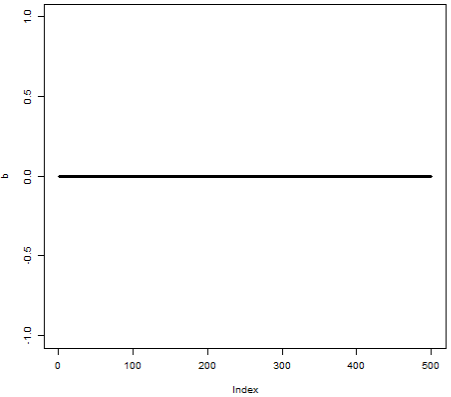
$$\hat{b}^{t+1} = \hat{b}^{t} + 2n^{-1}r\left(X'\hat{e} - \lambda\hat{b}^{t}\right)$$

No matrix inversion, just multiplications

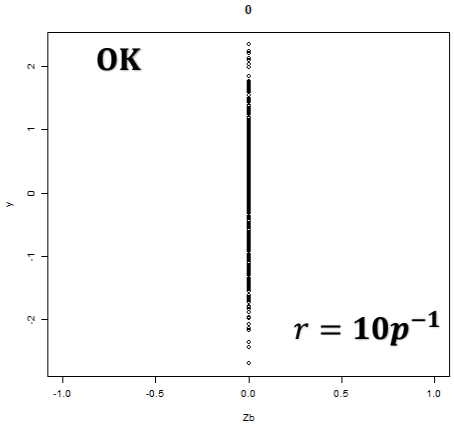
CORTEVA™ agriscience
Alencar.Xavier@Corteva.com
Corteva Biostatistics, Methods group

Gradient descent of a ridge regression

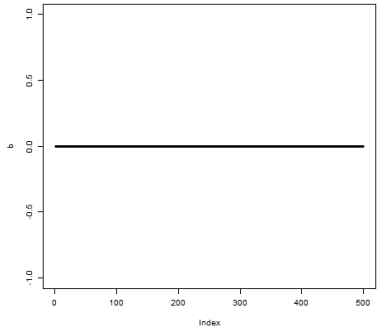
Marker effects

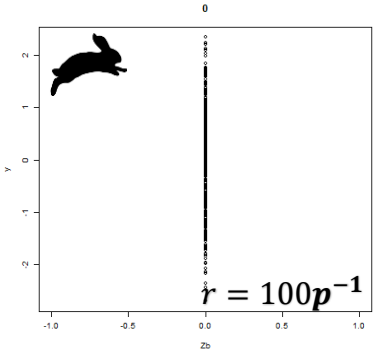


Fitness

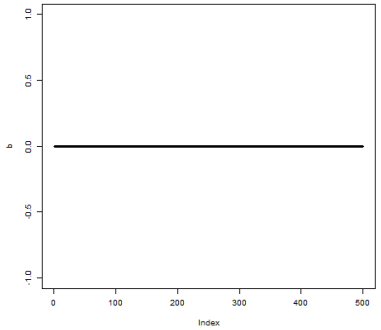


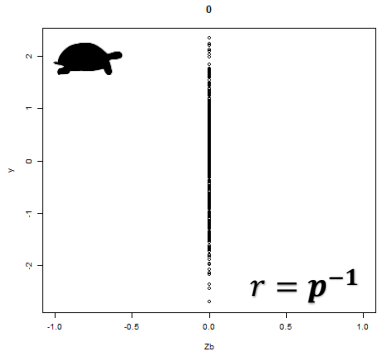
Large learning rate





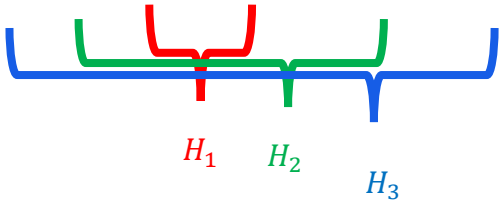
Small learning rate





Solution for neural networks

$$y = \alpha(\alpha(XB_1)B_2)b_3 + e$$



- **Fit layers**
  - $H_1 = \alpha(XB_1)$
  - $H_2 = \alpha(H_1B_2)$
  - $h_3 = H_2b_3$
- **Compute residuals for gradients**
  - $e_3 = y - h3$
  - $E_2 = \alpha(E_3B_3')$
  - $E_1 = \alpha(E_2B_2')$
- **Update coefficient**
  - $B_1^{t+1} = B_1^t - \gamma\left(\frac{2r_1}{n}[\,X'E_1 - \lambda B_1^t]\right)$
  - $B_2^{t+1} = B_2^t - \gamma\left(\frac{2r_2}{n}[\,H_1'E_2 - \lambda B_2^t]\right)$
  - $b_3^{t+1} = b_3^t - \gamma\left(\frac{2r_3}{n}[\,H_2'e_3 - \lambda b_3^t]\right)$

Scary? Top-to-bottom code ~ 30 lines

```
# DNN function
dnn = function(y, X, nit=1000, batch=250,RELU=FALSE, Leak=0.1,
               dropout=0, Lambda=0.1,LrnRate = 1,Nodes1=4, Nodes2=4){
  # Normalization
  if(is.null(ncol(y))) y = matrix(y)
  muY = colMeans(y,na.rm=T);  sdY = apply(y,2,sd,na.rm=T);  y = apply(y,2,scale)
  ActFun = tanh;  if(RELU) ActFun = function(x){x[x<0]=x[x<0]*Leak;return(x)}
  DropOut = function(x,prc=dropout){x[sample(length(x),length(x)*prc)]=0;return(x)}
  # Dimensions & Learning settings
  n = nrow(X);  p = ncol(X);  k = ncol(y)
  n1 = Nodes1; n2 = Nodes2;  lmb = Lambda; rate = LrnRate/c(p,n1,n2)
  b1 = matrix(rnorm(n1*p,0,1/p),p,n1)
  b2 = matrix(rnorm(n1*n2,0,1/n1),n1,n2)
  b3 = matrix(rnorm(n2,0,1/n2),n2,k)
  CNV1 = CNV2 = c()
  for(i in 1:nit){ # Iterations (backprop)
    if((i+1)%%100==1) cat('Iteration',i,'\n') # Sample batch
    w = sample(n,batch,replace=T);  y0 = y[w,]; X0 = X[w,]
    # Fit hidden layers (H) and Gradients (E)
    H1 = ActFun(X0%*%b1); H2 = ActFun(H1%*%b2); H3 = H2%*%b3
    e3 = y0-H3; if(anyNA(e3)) e3[is.na(e3)]=0
    e2 = ActFun(e3 %*% t(b3));  e1 = ActFun(e2 %*% t(b2))
    # Update coefficients
    b1 = b1 + DropOut(t(X0)%*%(e1)-lmb*b1)*(2/n)*rate[1]
    b2 = b2 + DropOut(t(H1)%*%(e2)-lmb*b2)*(2/n)*rate[2]
    b3 = b3 + DropOut(t(H2)%*%(e3))*(2/n)*rate[3]
    CNV1 = c(CNV1,mean(apply(e3,2,var,na.rm=T)))
    CNV2 = c(CNV2,mean(diag(cor(H3,y0,use='p')))) }
  out = list(af=ActFun,b1=b1,b2=b2,b3=b3,conv_MSE=CNV1,conv_GOF=CNV2,mu=muY,sd=sdY)
  class(out) = 'smalldnn'; return(out)}

predict.smalldnn = function(object,newdata){
  x = object; h = x$af(x$af(newdata%*%x$b1)%*%x$b2)%*%x$b3
  for(i in 1:ncol(h)) h[,i] = x$mu[i] + x$sd[i]*h[,i]
  return(h)}
```


CORTEVA™ agriscience

Alencar.Xavier@Corteva.com
Corteva Biostatistics, Methods group

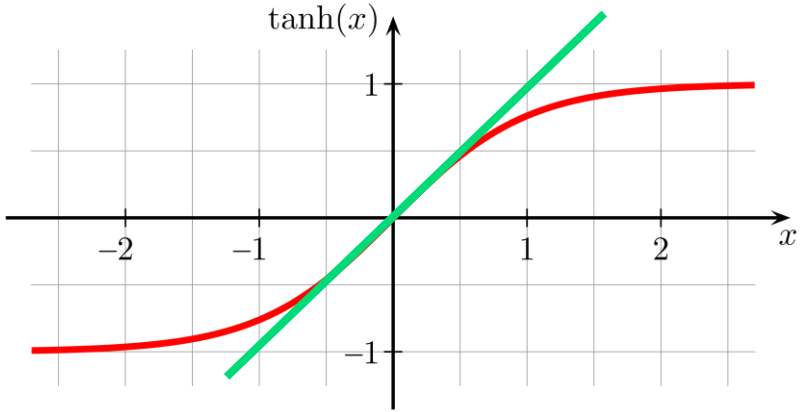
DNN jargons and nuisances

- Latent spaces = non-linear principal components
- “Nodes” = number of PCs = columns of H1 and H2 (driven by the # of columns of B1 and B2)

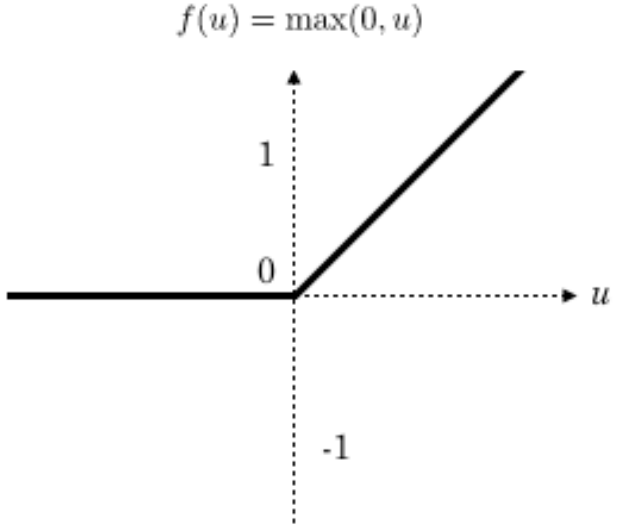
- **Adaptative momentum** speed up convergence: $B^{t+1} = B^t - \gamma\nabla^t - m\gamma\nabla^{t-t}$
- **Lazy loading**: Each iteration (update of B) uses a different chunk of data
- **Dropoff**: In each iteration, ignore parameters at random to mitigate overfit

- Coefficients **must** start with random values, e.g., $b \sim N(0, p^{-1})$
- **No guarantees** of similar results even if you fit the same model twice on the same data

Activation functions (α)



Linear between -0.5 and 0.5 ?
(most coefficients sit in this intervals)



4. Tree ensembles

$$y = n_t^{-1} \sum T(X) + e$$

- Predictions are averages from several ‘haplotypes’ of random markers
- Can capture complex patterns; Deal with large datasets;
- Requires *some* tuning; Not objectively interpretable;

Solution for tree ensembles

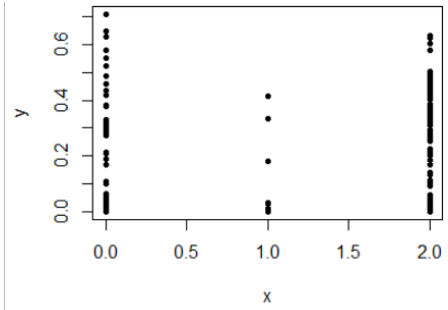
- No algebra. Tree methods are fully algorithmic.

- Three components
  - 1) Recursive partitioning (RP) = Function that splits the data
  - 2) Tree building = Function that runs, organize and store the RPs
  - 3) Ensemble method = Function that runs, organize and store trees

Recursive partitioning

- Univariate operation… $y = f(x)$, where x is a SNP coded as 012

```
data(tpod,bWGR)
x = gen[,100]
plot(x,y,pch=20)
```



Splits

$x \geq 1$

A/HB

$x > 1$

AH/B

Get means

```
> tapply(y,x>=1,mean)
    FALSE      TRUE
0.1194281 0.1976073
```

```
> tapply(y,x>1,mean)
    FALSE      TRUE
0.1210076 0.2016887
```

Compute error

$(y - \hat{y})'(y - \hat{y}) = 6.9998$

$(y - \hat{y})'(y - \hat{y}) = 6.9801$

Winner!

Outputs for SNP x:
Split rule: x>1
MSE: 6.9801

RP function:

- **Inputs**: x, y
- **Output**: best split rule, MSE

Tree-building

- Multiple responses… $y = f(x_1, x_2, x_3, \ldots, x_p)$

1) Run RP for all SNPs

	rule	mse
Gm03_46505146	1	7.263514
Gm06_17338681	2	6.999820
Gm09_1769811	1	7.166044
Gm11_1625959	1	7.264635
Gm13_42067890	2	7.274029
Gm17_26934954	2	7.292565
Gm19_2083595	2	7.281178

Rule 1 (A/HB): $x \geq 1$
Rule 2 (AH/B): $x > 1$

2) Identify winner, split y

Y1 = Y[$x \leq 1$]
Y2 = Y[$x > 1$]

3) Until reach stopping criteria:
Repeat 1 and 2 for each branch

Y1)

	rule	mse
Gm03_46505146	1	3.296907
Gm06_17338681	1	3.363726
Gm09_1769811	2	3.319813
Gm13_42067890	2	3.275088
Gm19_2083595	2	3.347615

Y1.1 = Y[$x \leq 1$]
Y1.2 = Y[$x > 1$]

Y2)

	rule	mse
Gm03_46505146	2	3.613022
Gm09_1769811	1	3.553694
Gm13_42067890	1	3.605200
Gm17_26934954	2	3.611959
Gm19_2083595	1	3.603940

Y2.1 = Y[$x < 1$]
Y2.2 = Y[$x \geq 1$]

(…)

Tree function:

- **Inputs**: X, y
- **Output**: Store splits rules, averages of branches

Ensemble methods

- Trees are known as "weak learners"… addressed by averaging many trees

1) Bagging (*e.g.*, random forest) = multiple **independent** trees

- Fit multiple trees (~ 500) using random samples of observations (bootstrapping or subsampling) and markers ($\sqrt{p}$ or $p/3$). The final predictor is the average of all trees.

2) Boosting (*e.g.*, xgboost, adaboost) = multiple **sequential** trees

- Boosting/Stacking: Fit a tree, use residuals to fit the next, and then next, and so on. Each tree may be fit different observations and markers. Final predictor is the sum of all trees.
- Partial Boosting: Fit a tree, reweight observations based on residuals (learning rate defines reweighting), fit the next tree, reweight, and so on. Final predictor is the average of all trees.

1. Introduction
2. Machines
   - Linear models
   - Kernel methods
   - Neural networks
   - Tree ensembles
3. **Validation**
   - Schemes and metrics
   - Information and case of study
4. Conclusion

Validation

CV schemes

- **Random CV** = Upper-bound predictive potential
- **Leave-one-out** = Assess structured scenarios (e.g., geography-out, year-out)
- **Holdout** = Reproduce true applications (e.g., predict individuals from upcoming)

	Genotype	Environment	Difficulty
CV00	New	New	*****
CV0	Observed	New	***
CV1	New	Observed	***
CV2	Observed	Observed	*

Adapted from Crossa et al. (2017) doi.org/10.1016/j.tplants.2017.08.011

Validation metrics

- **Correlations**
  - Most common metrics in breeding (e.g., predictability, accuracy when possible)
  - Pertinent to ranking and selection of complex traits
- **Prediction error**
  - Utilized when the predicted values must be as close as possible to original scale
  - Pertinent to risk prediction (e.g., disease risk)
- **Success**
  - Accommodate complex or subjective criteria, independent or otherwise
  - Pertinent to decision involving data from multiple sources (e.g., advancement)

Information

Information carried by each marker:

- ***Linkage disequilibrium (LD)***: Marker proximity to a causative locus or regions, irrespective of population source.
- ***Linkage:*** Marker inherited from a specific source or parents. Also referred to as: co-segregation, haplotype, short-term LD.
- ***Relationship***: Marker information attributed to population structure. Captures differences among families and populations.

NOTE: Whether allelic information is additive, dominant or epistatic depends on parametrization and coding.

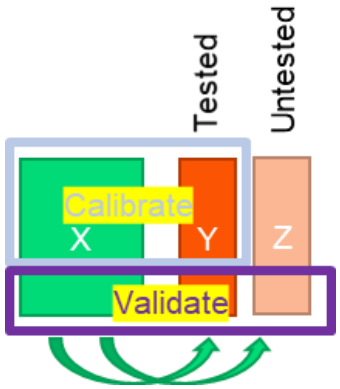
Information

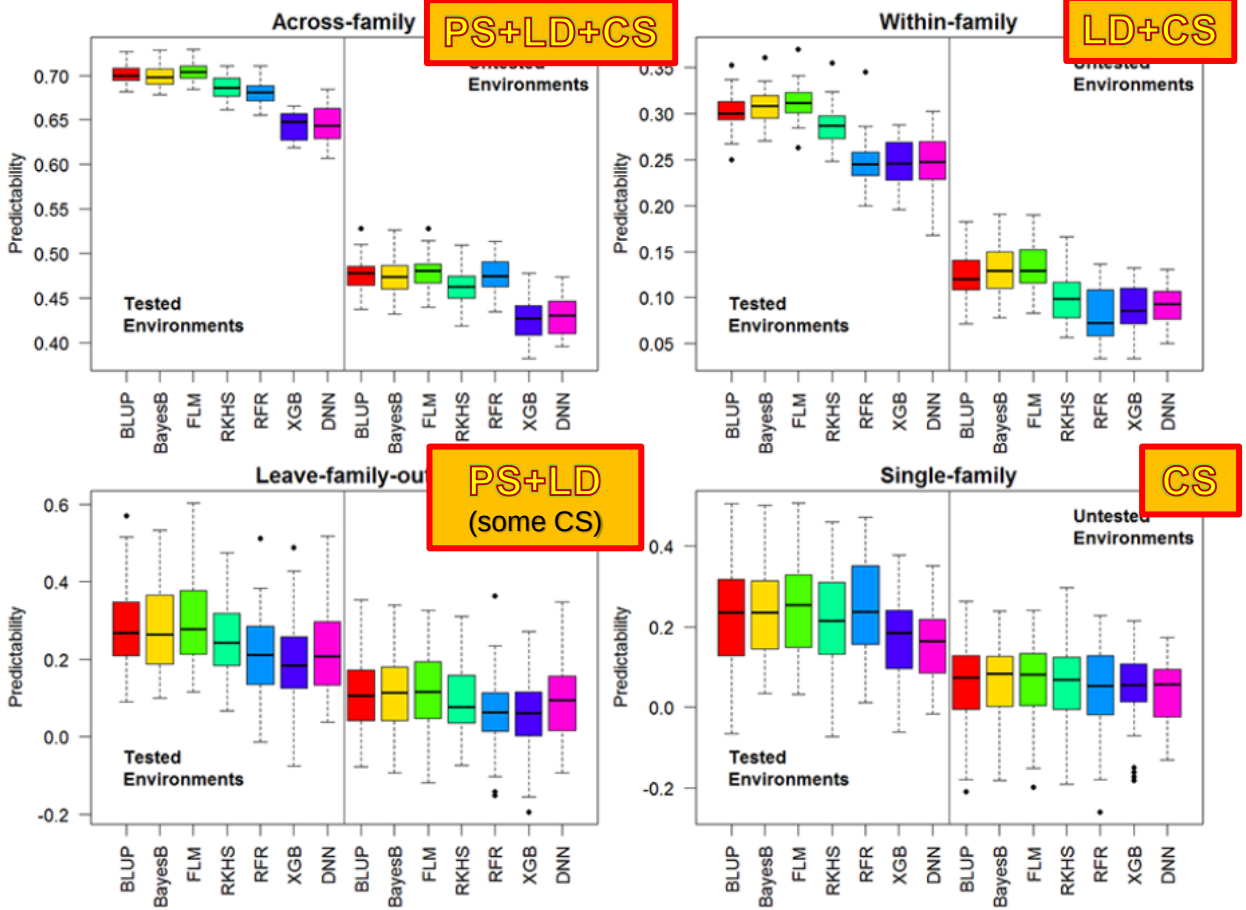
Genetic information assessed by cross-validation setup

- ***Intra-family***: Linkage*
- ***Within-family***: Linkage and LD
- ***Across-family***: Relationships**, Linkage and LD
- ***Leave-family-out***: Relationships and LD

- ***Untested environments***: Same as above + GxE component

CV scheme





SoyNAM data
ES: 2012 (7 loc)
PS: 2013 (4 loc)
#Fam = 40
Genos = 5600
SNPs = 4300
Obs: 3k-5k obs/loc

Type of information captured by SNP

- Population structure (PS)
- Linkage disequilibrium (LD)
- Cosegregation / Haplotype (CS)

Figure 1. Four cross-validation schemes illustrating predictability of various methods utilized for genomic prediction. Grain yield models from the SoyNAM population, validated upon unobserved individuals from tested and untested environments.

CORTEVA™ agriscience
Alencar.Xavier@Corteva.com
Corteva Biostatistics, Methods group

1. Introduction
2. Machines
   - Linear models
   - Kernel methods
   - Neural networks
   - Tree ensembles
3. Validation
   - Schemes and metrics
   - Information and case of study
4. **Conclusion**

Conclusion

Thank you for your attention!

Remarks:

1) There are discrepancies in thought and nomenclature between ML and QG
2) We reviewed the 4 major types of machines utilized in genomic predictions
3) Cross-validation help us to understand methods' strengths and limitations

Questions??

Alencar Xavier

Alencar.Xavier@Corteva.com

Acknowledgements: David Habier for extensive revision of the study; Radu Totir for supporting the publication.

Unnecessarily complex analysis should not be used as a foil to disguise lower quality datasets

Kruuk (2004 *apud* Walsh and Lynch 2018)

Data > Method