Introduction to Machine Learning in R

Nikolay Oskolkov, MRG Group Leader, LIOS, Riga, Latvia

R course, 05.02.2026

@NikolayOskolkov

@oskolkov.bsky.social

Personal homepage:
https://nikolay-oskolkov.com

Image generated by ChatGPT

Session content

Topics we'll cover in this session:

1) How statistics is different from machine learning

2) K-means clustering as fingerprint of machine learning

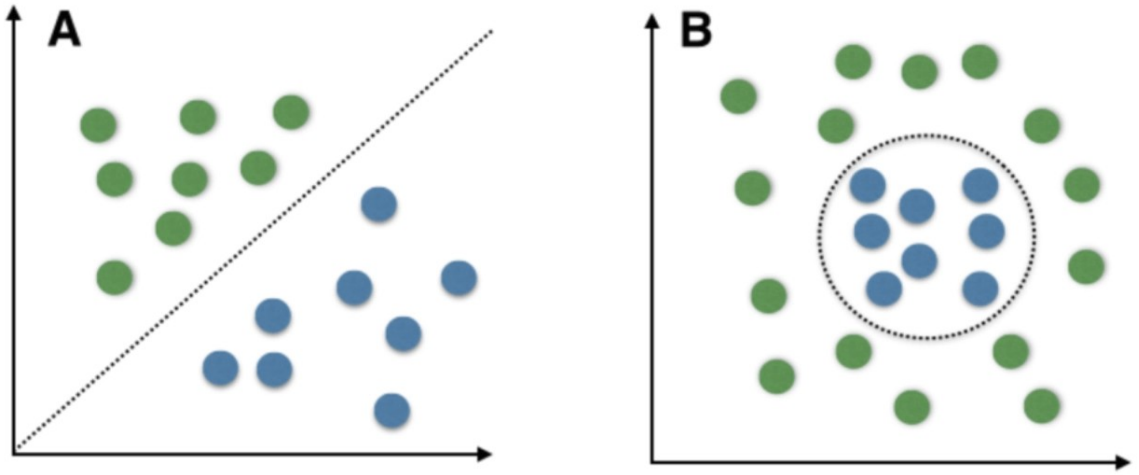
3) Basics of machine learning mentality

4) Linear machine learning: train, validation and test split

5) Overfitting vs. underfitting, coding gradient descent from scratch in R

Y = f (X), where X is input (data) and Y is output (response)

Y is present – supervised machine learning
Y is absent – unsupervised machine learning



A: Linearly Separable Data B: Non-Linearly Separable Data

Moving from statistics to machine learning

- Statistics is more **analytical** (pen & paper)

$$\mathrm{L}\left(\mathrm{x_i} \mid \mu, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp^{-\frac{\sum_{\mathrm{i}=1}^{\mathrm{N}} (\mathrm{x_i} - \mu)^2}{2\sigma^2}}$$

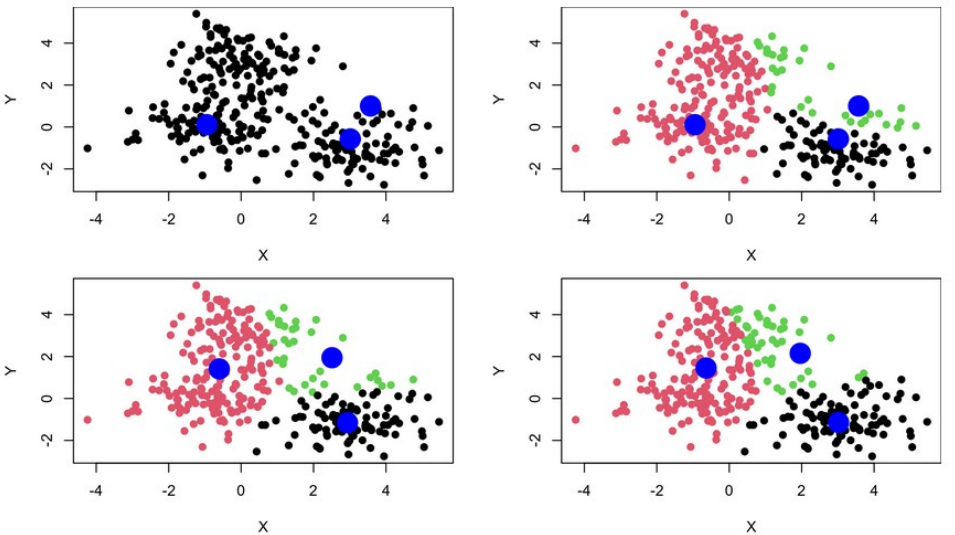
$$\frac{\partial \mathrm{L}\left(\mathrm{x_i} \mid \mu, \sigma^2\right)}{\partial \mu} = 0; \quad \frac{\partial \mathrm{L}\left(\mathrm{x_i} \mid \mu, \sigma^2\right)}{\partial \sigma^2} = 0$$

$$\mu = \frac{1}{N} \sum_{i=0}^{N} x_i \; - \text{mean estimator}$$

$$\sigma^2 = \frac{1}{N} \sum_{i=0}^{N} (x_i - \mu)^2 \; - \text{variance estimator}$$

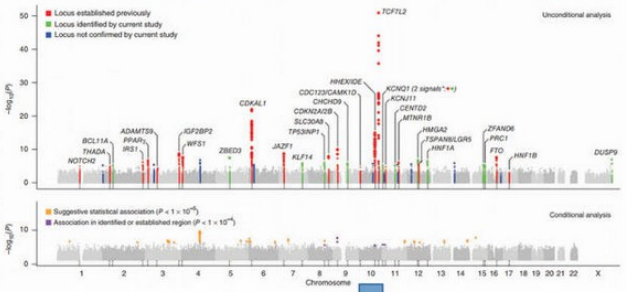
- Machine Learning is more **algorithmic** (ex. K-means)

```r
K = 3; set.seed(123); c = X[sample(1:dim(X)[1],K),]; par(mfrow=c(2,2),mai=c(0.8,1,0,0))
plot(X, xlab = "X", ylab = "Y", pch = 19); points(c, col = "blue", cex = 3, pch = 19)
for(t in 1:3)
{
  l <- vector()
  for(i in 1:dim(X)[1])
  {
    d <- vector(); for(j in 1:K){d[j] <- sqrt((X[i,1]-c[j,1])^2 + (X[i,2]-c[j,2])^2)}
    l[i] <- which.min(d)
  }
  plot(X, xlab="X", ylab="Y", col=l, pch=19); points(c, col="blue", cex=3, pch=19)
  s = list(); for(i in unique(l)){s[[i]] <- colMeans(X[l==i,])}; c = Reduce("rbind", s)
}
```



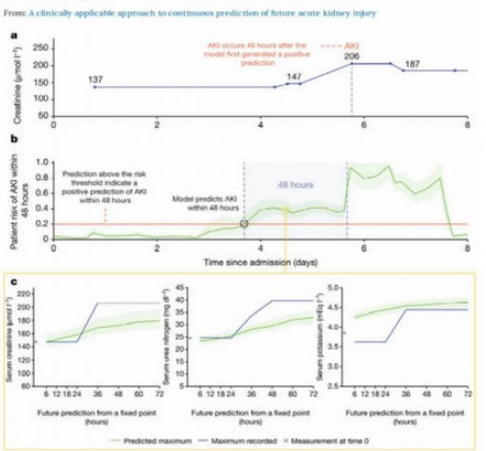
Statistics vs. machine learning: prediction

Statistics searches for candidates

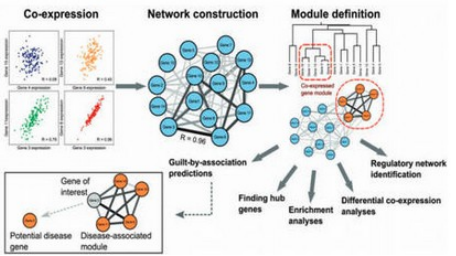


Consequence

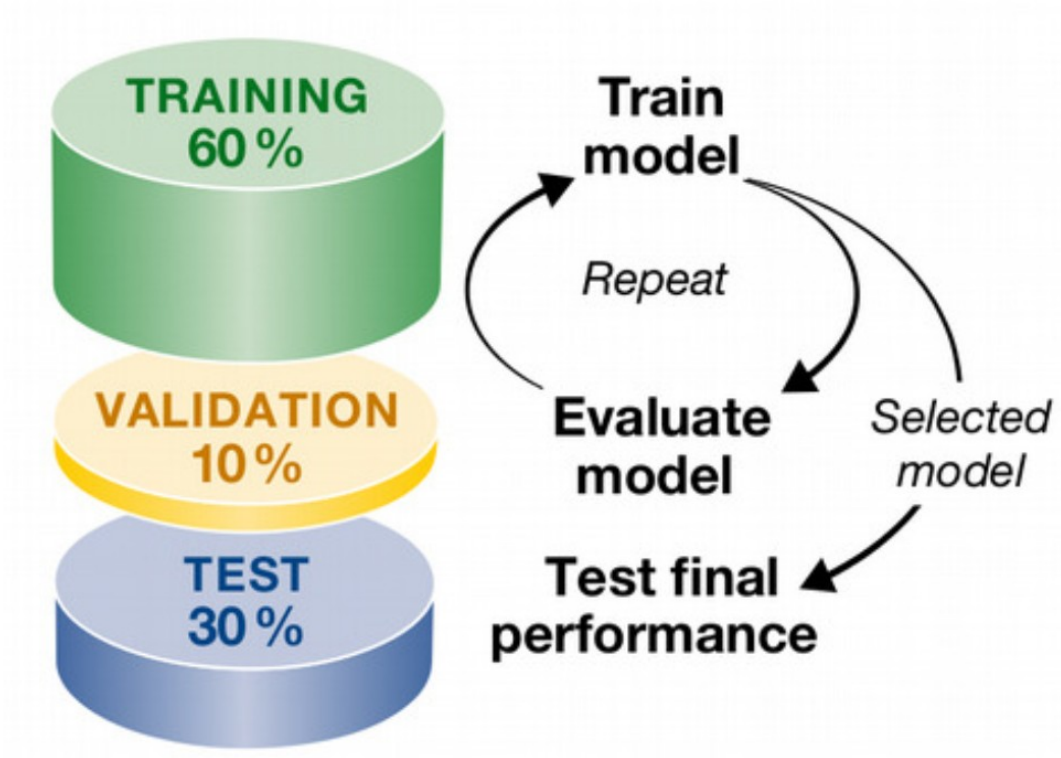
Machine Learning optimizes prediction



Consequence



How does machine learning work?



Machine Learning typically involves five basic steps:

1. Split data set into **train**, **validation** and **test** subsets

2. Fit the model on the train subset

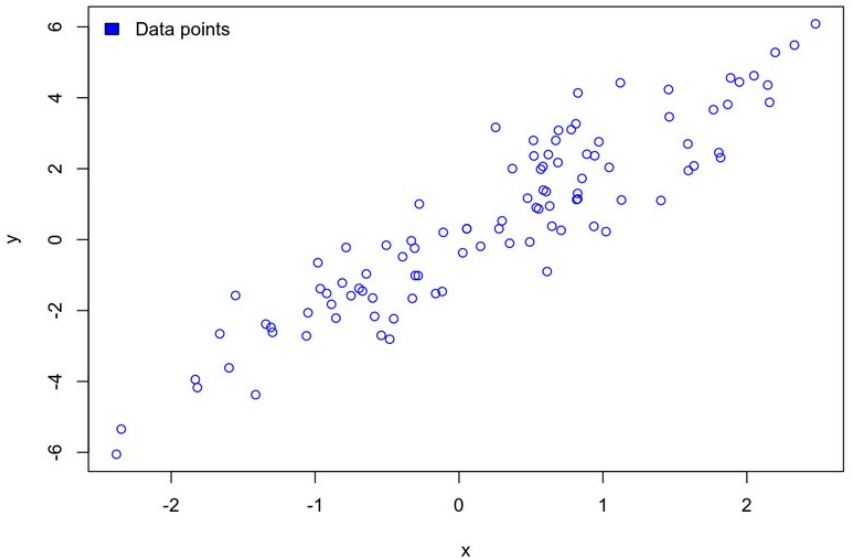
3. Validate your model on the validation subset

4. Repeat train - validation split many times and tune **hyperparameters**

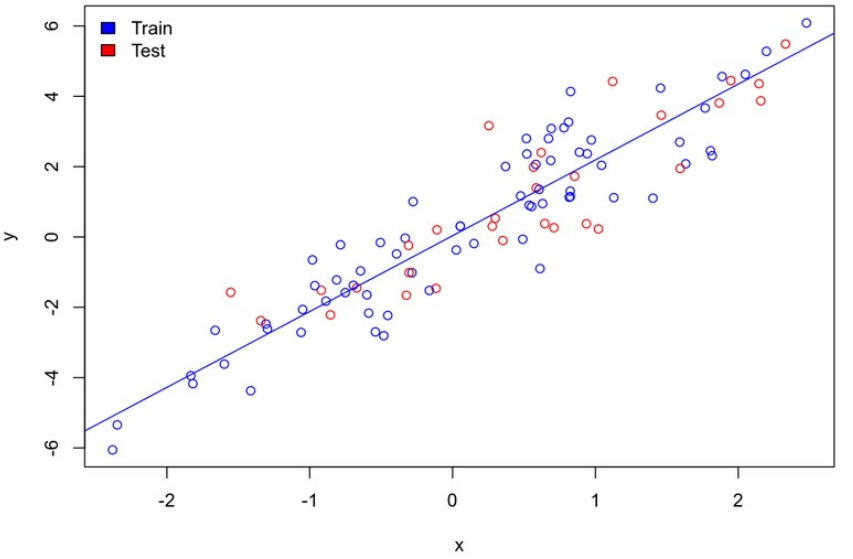
5. Test the accuracy of the optimized model on the test subset.

Toy example of machine learning

```r
N <- 100
x <- rnorm(N)
y <- 2 * x + rnorm(N)
df <- data.frame(x, y)
plot(y ~ x, data = df, col = "blue")
legend("topleft", "Data points", fill = "blue", bty = "n")
```

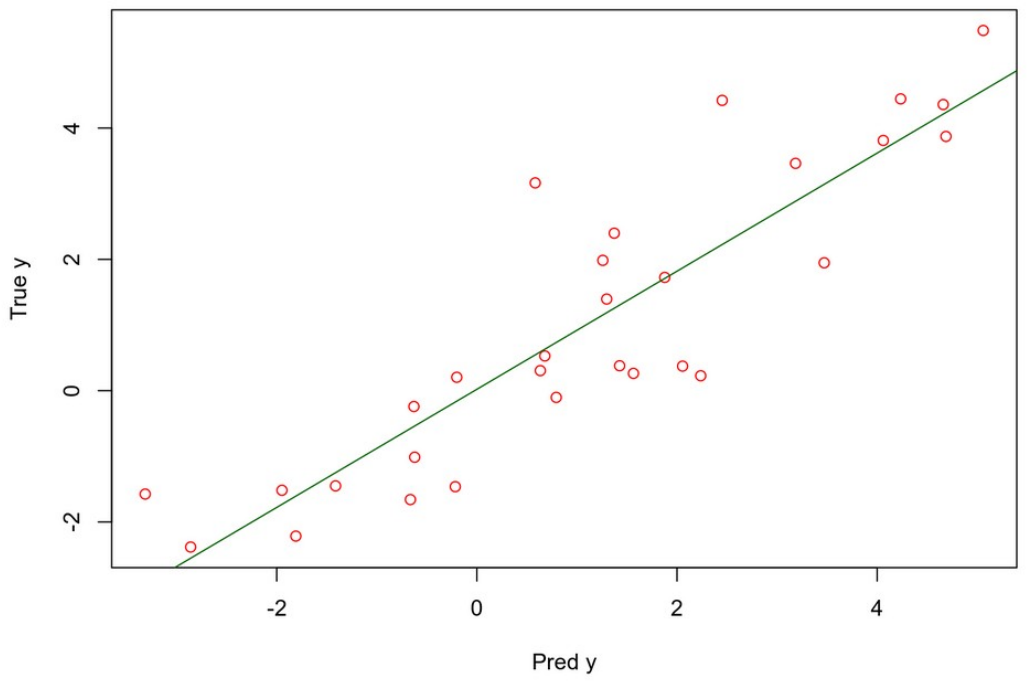



```r
train <- df[sample(1:dim(df)[1], 0.7 * dim(df)[1]), ]
test <- df[!rownames(df) %in% rownames(train), ]
df$col <- ifelse(rownames(df) %in% rownames(test), "red", "blue")
plot(y ~ x, data = df, col = df$col)
legend("topleft", c("Train","Test"), fill=c("blue","red"), bty="n")
abline(lm(y ~ x, data = train), col = "blue")
```



Toy example: model validation

```r
test_predicted <- as.numeric(predict(lm(y ~ x, data = train), newdata = test))
plot(test$y ~ test_predicted, ylab = "True y", xlab = "Pred y", col = "red")
abline(lm(test$y ~ test_predicted), col = "darkgreen")
```

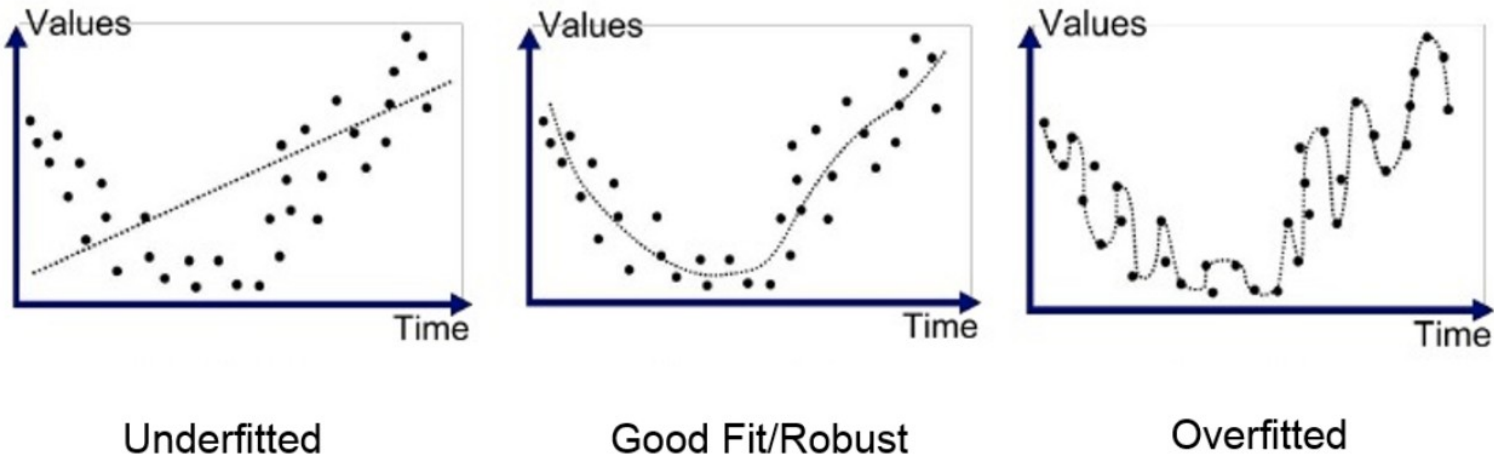



```r
summary(lm(test$y ~ test_predicted))
```

```
Call:
lm(formula = test$y ~ test_predicted)

Residuals:
     Min       1Q   Median       3Q      Max
-1.80597 -0.78005  0.07636  0.52330  2.61924

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)     0.02058    0.21588   0.095    0.925
test_predicted  0.89953    0.08678  10.366 4.33e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
' ' 1

Residual standard error: 1.053 on 28 degrees of freedom
Multiple R-squared:  0.7933,    Adjusted R-squared:
0.7859
F-statistic: 107.4 on 1 and 28 DF,  p-value: 4.329e-11
```

Thus the model explains 79% of variation on the test subset.

Underfitting vs. Overfitting

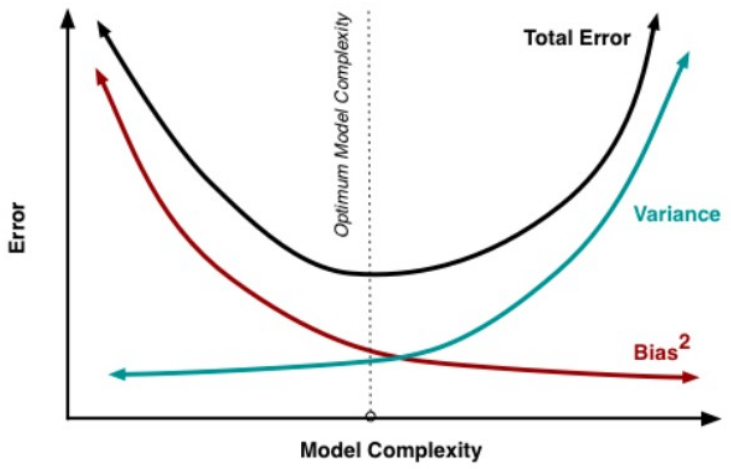


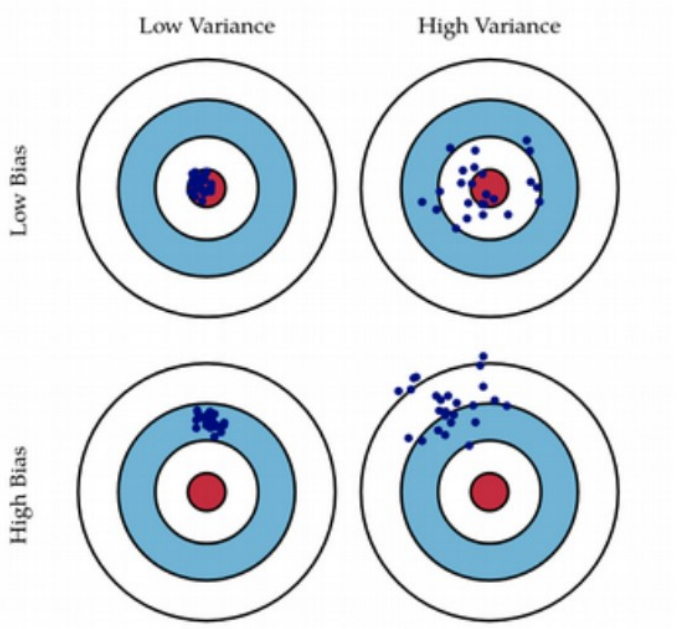
- Akaike Information Criterion (AIC):

$$\mathrm{AIC} = 2\mathrm{k} - 2\ln(\mathrm{L})$$

- Random Forest: each tree overfitted, but ensemble of trees performs very well

Bias-Variance Tradeoff





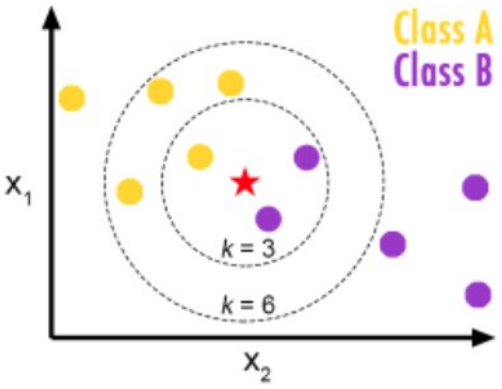
$$Y = f(X) \Longrightarrow \text{Reality}$$

$$\mathrm{Y} = \hat{\mathrm{f}}\,(\mathrm{X}) + \text{Error} \Longrightarrow \text{Model}$$

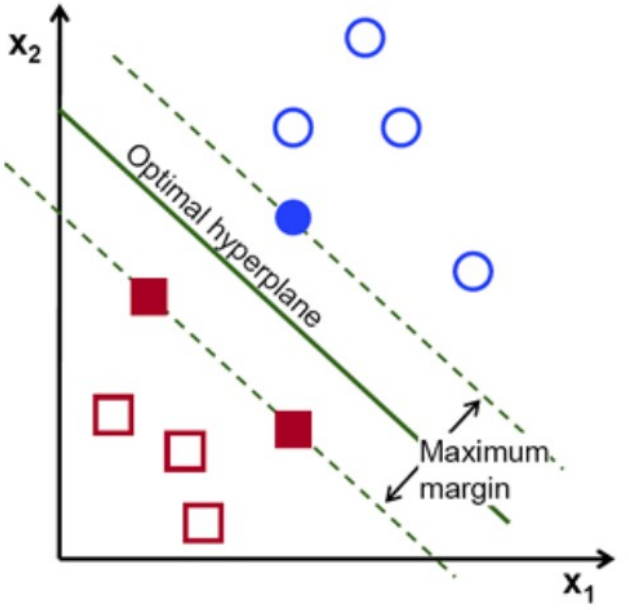
$$\text{Error}^2 = (\mathrm{Y} - \hat{\mathrm{f}}\,(\mathrm{X}))^2 = \text{Bias}^2 + \text{Variance}$$

- It is mathematically proven that ensemble learning keeps the Bias the same but leads to a large decrease of Variance

KNN and SVM

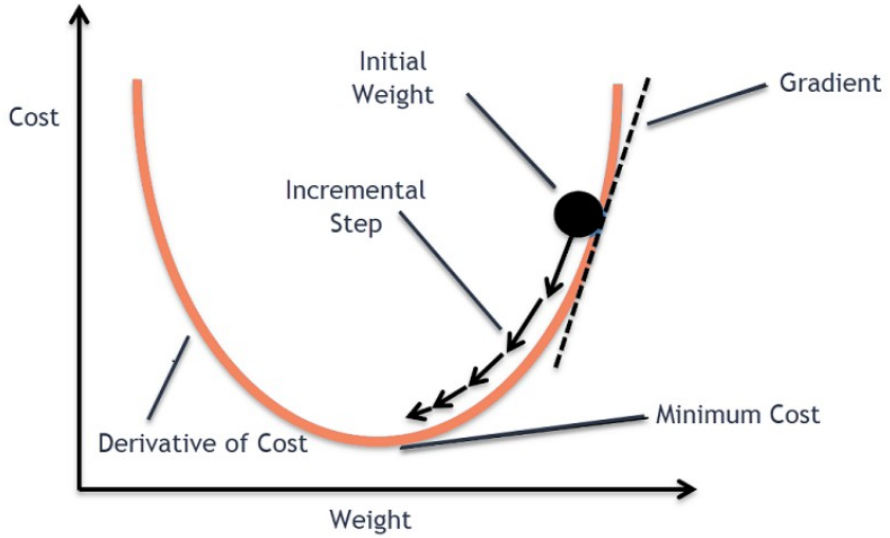


- How many out of K neighbors belong to each class
- Majority voting
- Non-linear



- Draw hyperplane that separates classes
- Maximize margins
- Can be linear and non-linear

Gradient descent



$$\hat{\alpha}, \hat{\beta} = \operatorname{argmin} \mathrm{E}(\alpha, \beta)$$

$$\frac{\partial E(\alpha,\beta)}{\partial \alpha} = -\frac{2}{n}\sum_{i=1}^{n}(y_i - \alpha - \beta x_i)$$

$$\frac{\partial E(\alpha,\beta)}{\partial \beta} = -\frac{2}{n}\sum_{i=1}^{n} x_i(y_i - \alpha - \beta x_i)$$

Numeric implementation of gradient descent:

$$y_i = \alpha + \beta x_i + \epsilon,\ i = 1 \ldots n$$

$$E(\alpha,\beta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \alpha - \beta x_i)^2$$

$$\alpha_{i+1} = \alpha_i - \eta \frac{\partial E(\alpha,\beta)}{\partial \alpha}\bigg|_{\alpha=\alpha_i,\beta=\beta_i}$$

$$\beta_{i+1} = \beta_i - \eta \frac{\partial E(\alpha,\beta)}{\partial \beta}\bigg|_{\alpha=\alpha_i,\beta=\beta_i}$$

Coding gradient descent from scratch in R

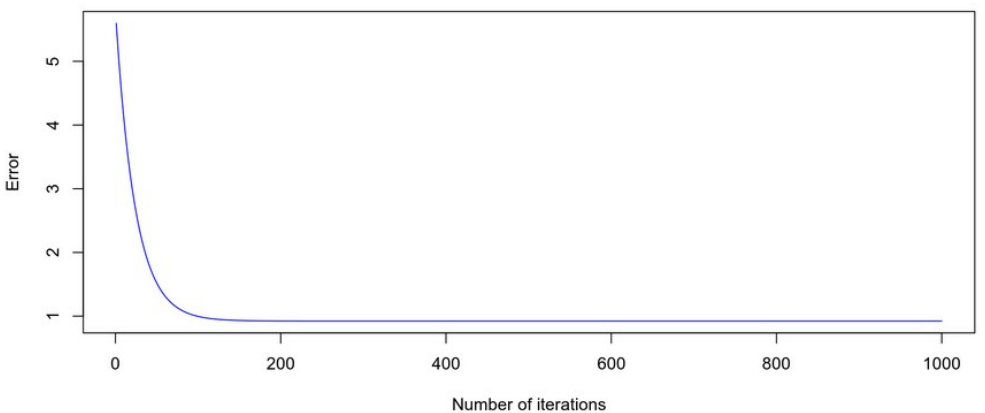
```r
n <- 100 # sample size
x <- rnorm(n) # simulated expanatory variable
y <- 3 + 2 * x + rnorm(n) # simulated response variable
summary(lm(y ~ x))
```

```
Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-1.9073 -0.6835 -0.0875  0.5806  3.2904

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.89720    0.09755   29.70   <2e-16 ***
x            1.94753    0.10688   18.22   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9707 on 98 degrees of freedom
Multiple R-squared:  0.7721,    Adjusted R-squared:  0.7698
F-statistic:   332 on 1 and 98 DF,  p-value: < 2.2e-16
```

Let us now reconstruct the intercept and slope from gradient descent

```r
alpha <- vector(); beta <- vector()
E <- vector(); dEdalpha <- vector(); dEdbeta <- vector()
eta <- 0.01; alpha[1] <- 1; beta[1] <- 1 # initialize alpha and beta
for(i in 1:1000)
{
  E[i] <- (1/n) * sum((y - alpha[i] - beta[i] * x)^2)
  dEdalpha[i] <- - sum(2 * (y - alpha[i] - beta[i] * x)) / n
  dEdbeta[i] <- - sum(2 * x * (y - alpha[i] - beta[i] * x)) / n

  alpha[i+1] <- alpha[i] - eta * dEdalpha[i]
  beta[i+1] <- beta[i] - eta * dEdbeta[i]
}
print(paste0("alpha = ", tail(alpha, 1),", beta = ", tail(beta, 1)))
```

```
[1] "alpha = 2.89719694937354, beta = 1.94752837381973"
```



Session summary

Take home messages of the session:

1) Machine Learning is different from statistics in terms of more algorithmic approach and prediction

2) Train, validation and test are fundamental machine learning steps that aim at improving model generalizability

3) Challenge in machine learning is to find a balance between overfitting and underfitting

4) Gradient descent is the core engine of machine learning and is based on computing derivatives of the loss function

Acknowledgments: LIOS + TARGETWISE

Latvian Institute of Organic Synthesis

2027 National Development Plan

Funded by the European Union