EpiMed Open Course – Session 4

AI for Omics – Use case of leukemia classification – Part 1

Ekaterina Flin

13/04/2020

Plan

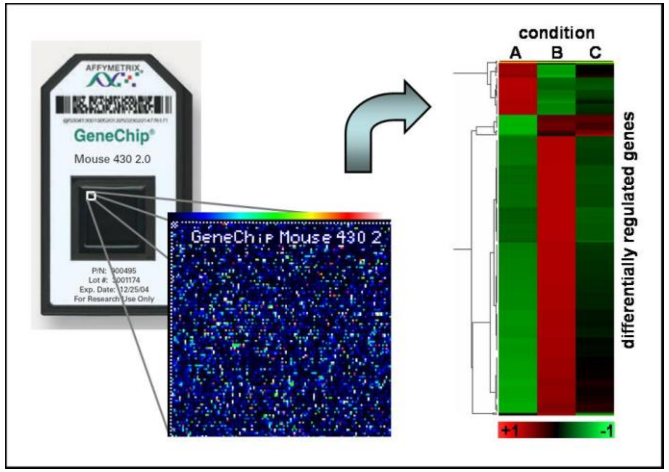
Part 1: Data visualization and preprocessing

- Challenges in omics data and possible solutions
- Methods for data visualization
- Variable reduction
- Model generalization

Part 2: Model training

- Training of 5 different models (linear and non-linear)
- Benchmark

Omics data



Microarrays

New Generation Sequencing (NGS)

- Genomics

  genetic variants, insertions, deletions, …
  10 GB

- Transcriptomics

  expression levels of genes
  10 MB – 10 GB

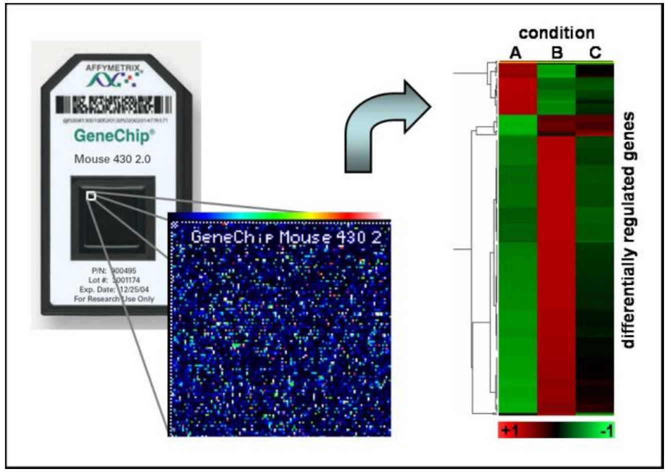
- Proteomics

  protein presence in a certain cell type
  10 MB

- Epigenomics

  DNA methylation, histone modifications, …
  10 MB – 10 GB

Omics data



Microarrays

New Generation Sequencing (NGS)

- Genomics
  genetic variants, insertions, deletions, …
  10 GB

- Transcriptomics
  expression levels of genes
  10 MB – 10 GB

- Proteomics
  protein presence in a certain cell type
  10 MB

- Epigenomics
  DNA methylation, histone modifications, …
  10 MB – 10 GB

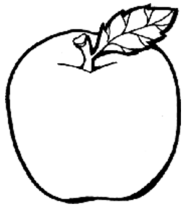
Classes of Learning Problems

Supervised Learning

Data: (x, y)
x is data, y is label

Goal:
Learn function to map
x → y

Apple example:



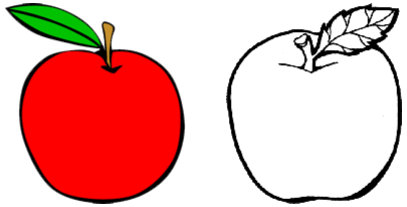
This thing is an apple.

Unsupervised Learning

Data: x
x is data, no labels!

Goal:
Learn underlying structure

Apple example:



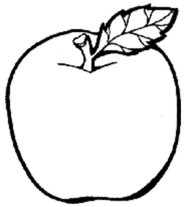
This thing is like the other thing.

Reinforcement Learning

Data: state-action pairs

Goal:
Maximize future rewards over many time steps

Apple example:



Eat this thing because it will keep you alive.

Source: MIT Course 6.S191 Introduction to Deep Learning

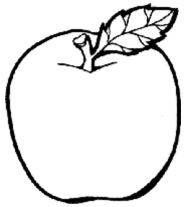
Classes of Learning Problems

Supervised Learning

Data: (x, y)
x is data, y is label

Goal:
Learn function to map
x → y

Apple example:



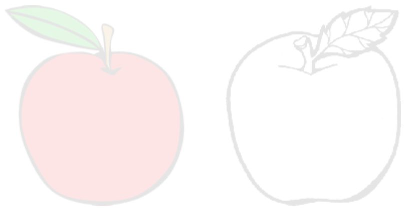
This thing is an apple.

Unsupervised Learning

Data: x
x is data, no labels!

Goal:
Learn underlying
structure

Apple example:



*This thing is like
the other thing.*

Reinforcement Learning

Data: state-action pairs

Goal:
Maximize future rewards
over many time steps

Apple example:

*Eat this thing because it
will keep you alive.*

Source: MIT Course 6.S191 Introduction to Deep Learning

Supervised Machine Learning

Linear classifiers:

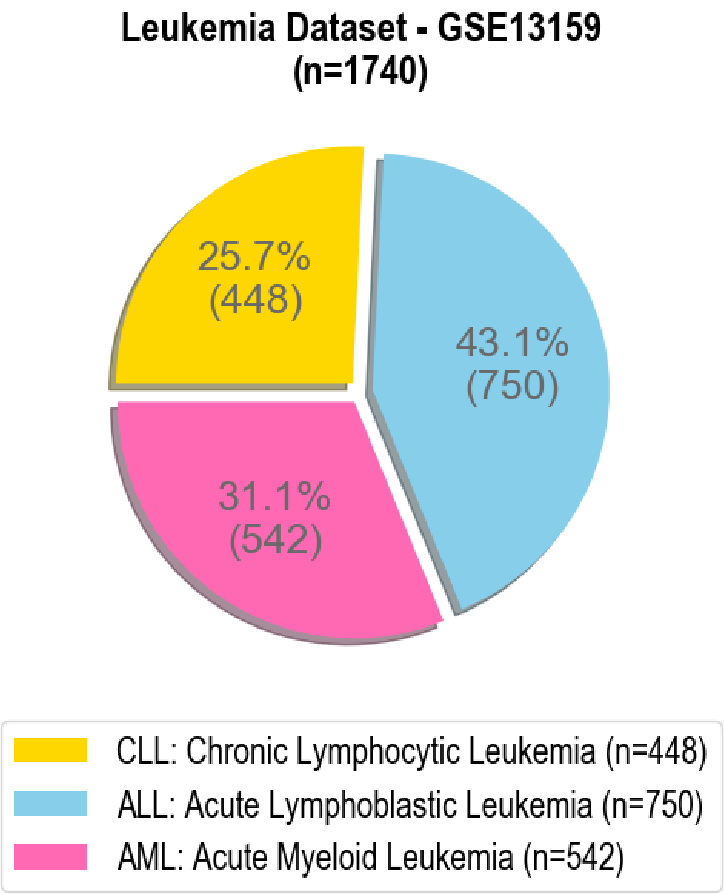
- Logistic Regression
- Linear Support Vector Machine (SVM)

Non-linear classifiers:

- SVM with a non-linear kernel
- Random Forest
- Neural Network

Use case: leukemia dataset

Goal: Predict leukemia type CLL, ALL or AML



Transcriptomic data, microarrays

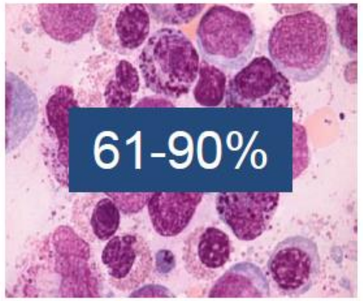
Total number of samples = 1 740

Total number of genes (“features”) = 21 875

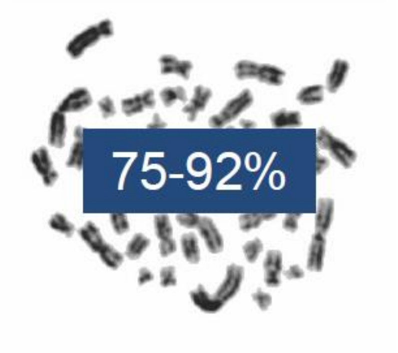
3 labels to identify : CLL, ALL or AML

Use case: leukemia dataset

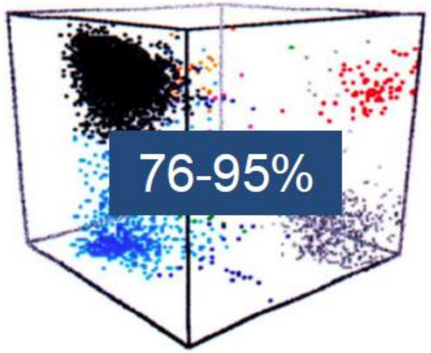
Accuracy of diagnosis in leukemia with currently used techniques: **61-95%**.
Can we do better by using transcriptomic data?



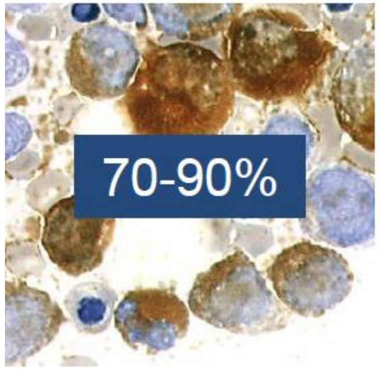
Morphology



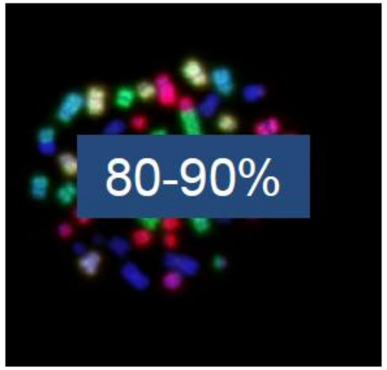
Cytogenetics



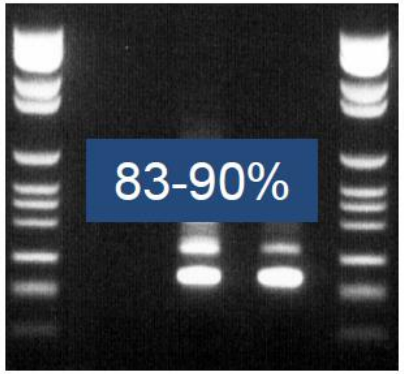
Immunophenotyping



Cytochemistry



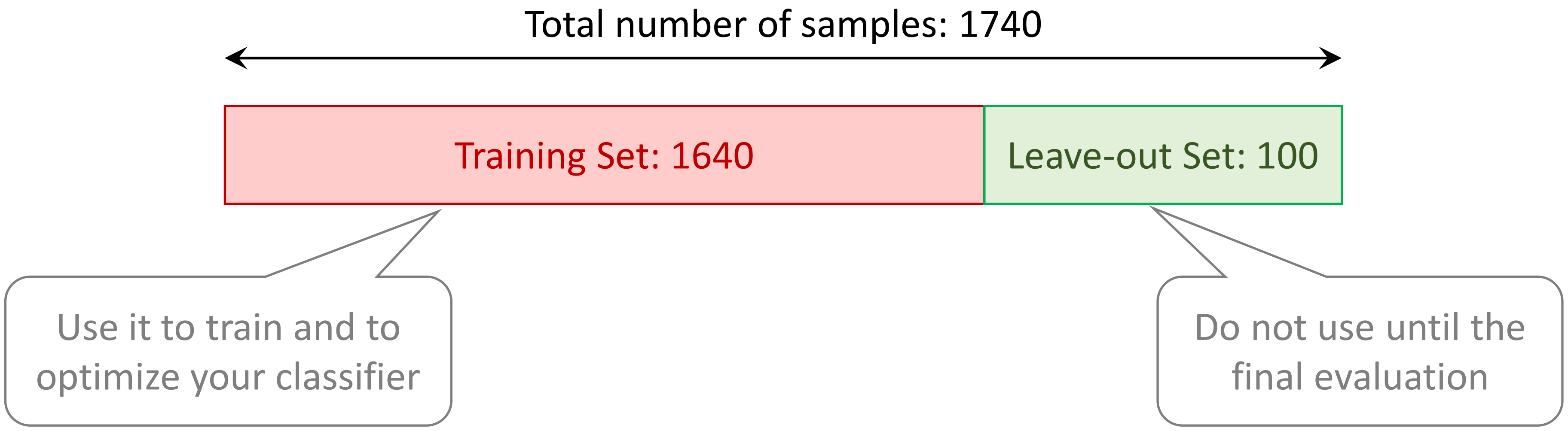
FISH



PCR

Source: K. Mills, The American Society of Hematology, 2007.

Use case: leukemia dataset



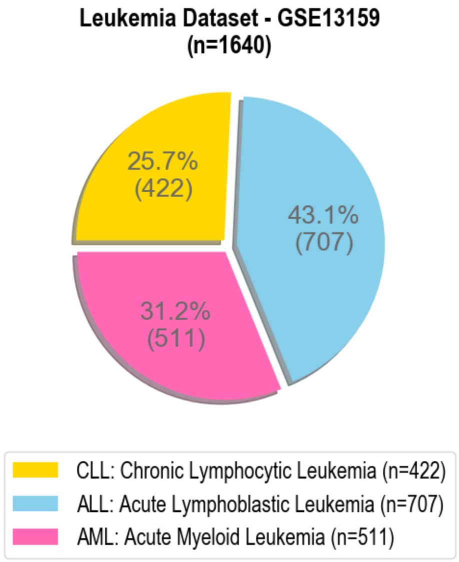
Total number of samples: 1740

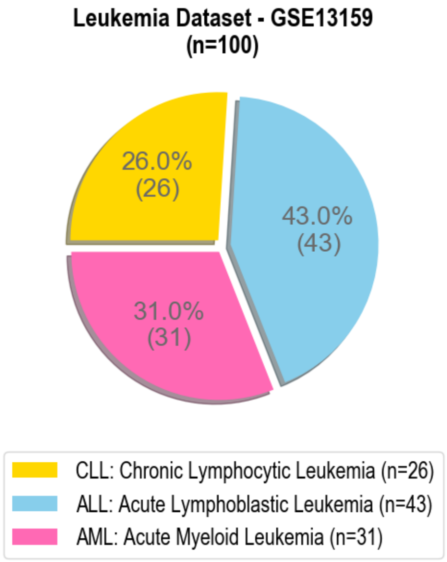
Training Set: 1640

Leave-out Set: 100

Use it to train and to optimize your classifier

Do not use until the final evaluation

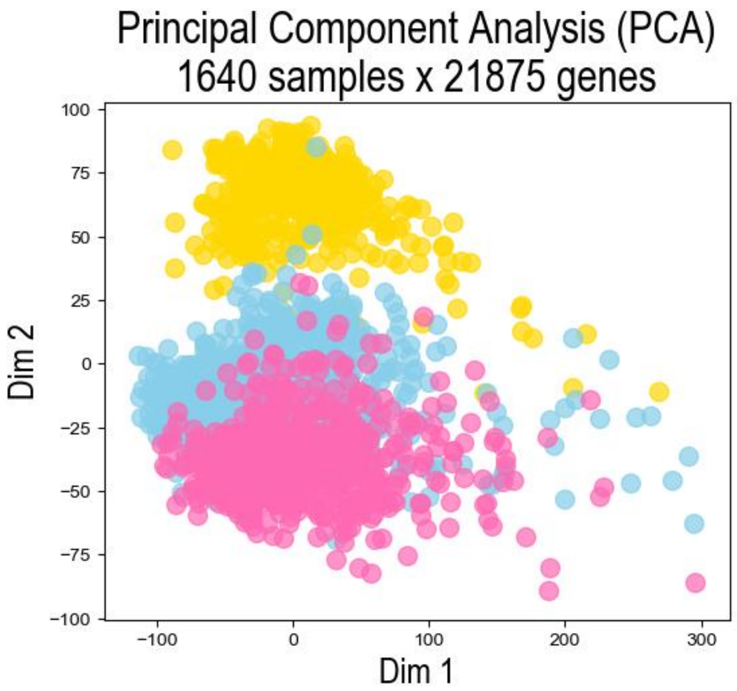




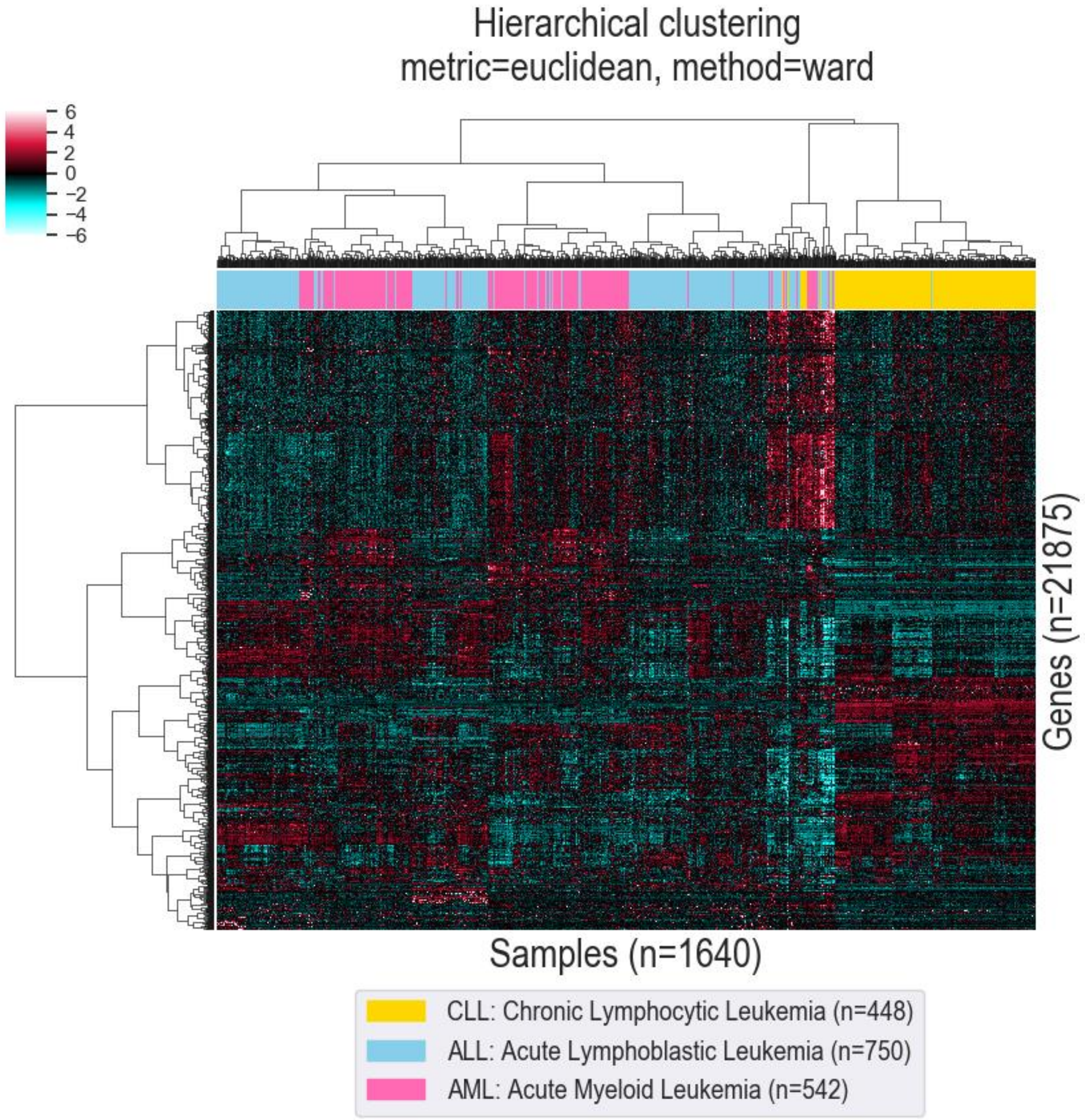
Visualization of data: Unsupervised Machine Learning

Training Set: 1640

Use the training set only



Principal Component Analysis (PCA)
1640 samples x 21875 genes

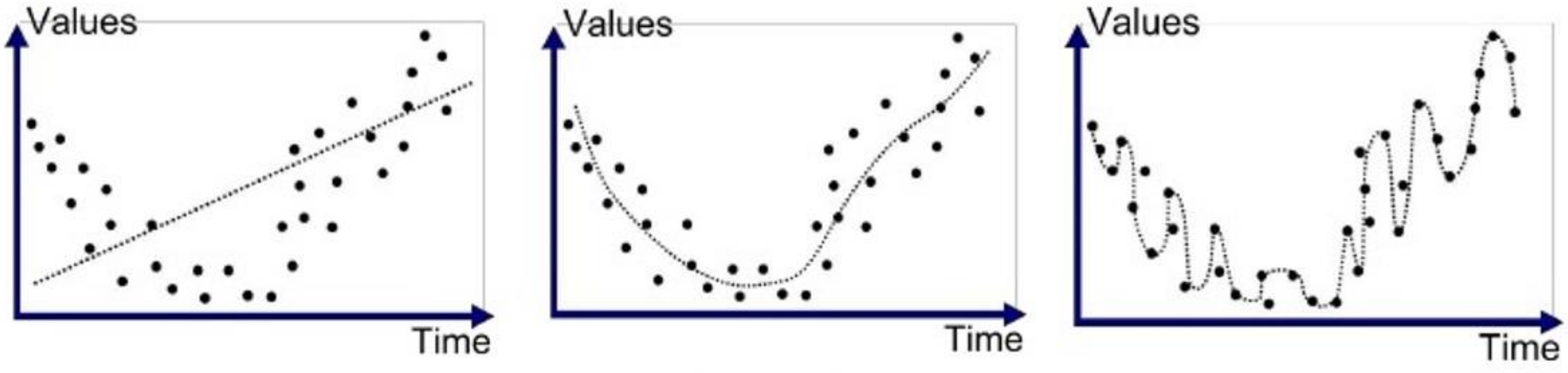


Hierarchical clustering
metric=euclidean, method=ward

Problem N1: Robustness and generalization

- Results obtained in one study may not work in another independent study (poor generalization)

- We need to develop robust approaches that insure validation in other datasets, or at least increase our chances for validation



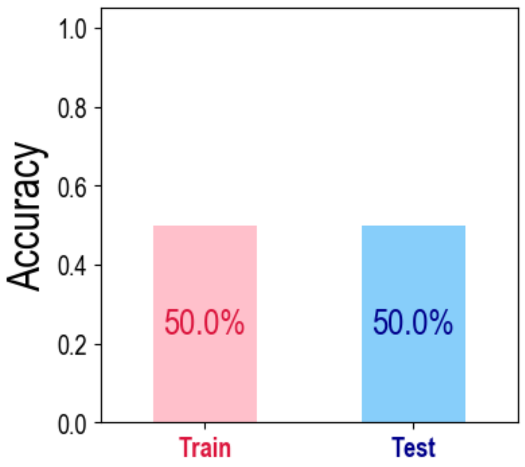
Underfitted

Good Fit/Robust

Overfitted

Problem N1: Robustness and generalization

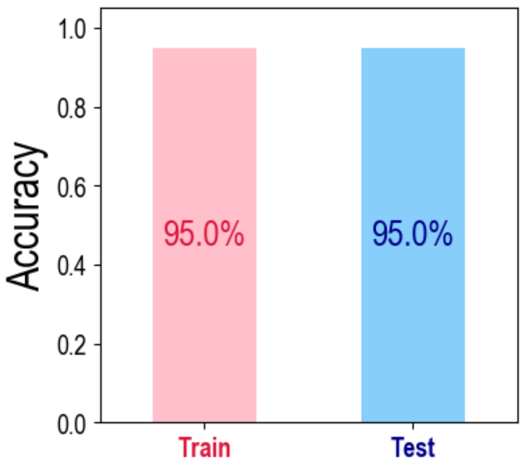
Underfitted



The classifier does not perform well in both training and test datasets.

The prediction is not better than just an arbitrary guess (without any model training).

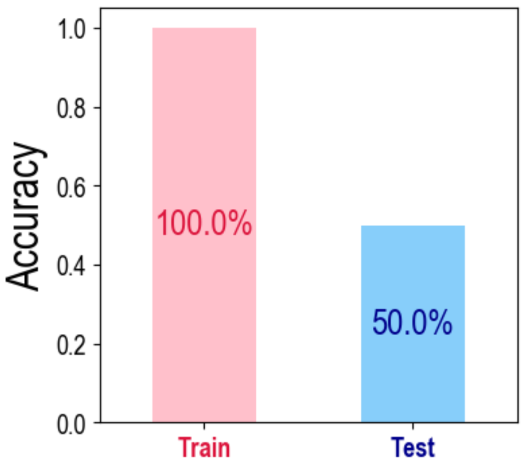
Robust



The classifier has strong generalization properties.

It gives similar prediction accuracy in both training and test datasets.

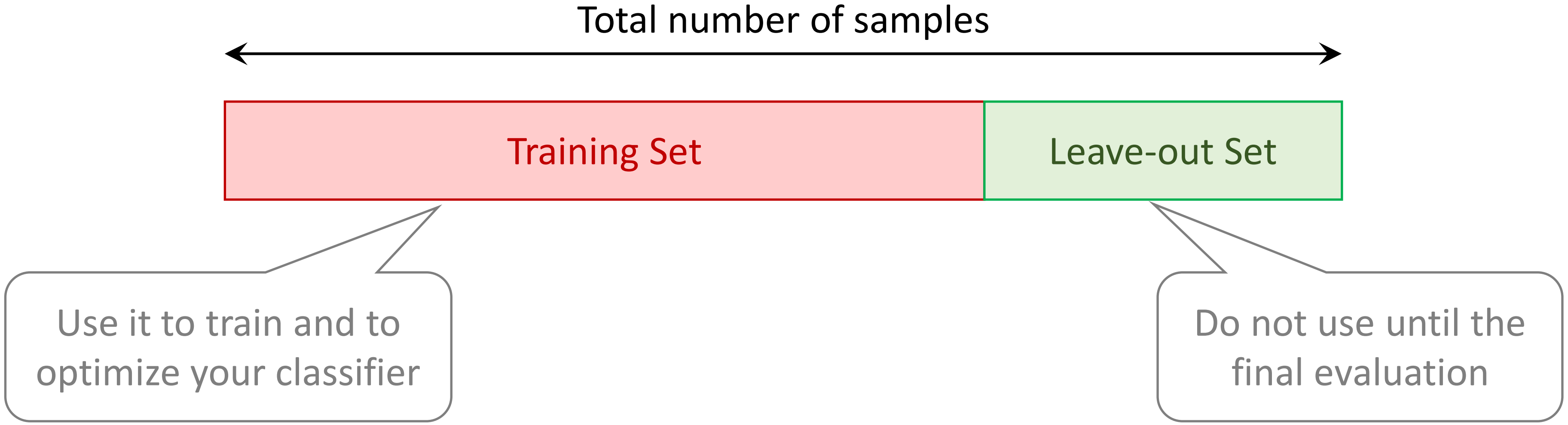
Overfitted



The classifier memorizes particular data points in training dataset, instead of learning from data.

This model can not be generalized on test dataset.

Problem N1: Robustness and generalization



Total number of samples

Training Set

Leave-out Set

Use it to train and to optimize your classifier

Do not use until the final evaluation

K-fold cross-validation			Accuracy of prediction in test dataset
Train	Train	Test	70%
Test	Train	Train	80%
Train	Test	Train	90%

Cross-validation in Python

```
from sklearn.model_selection import StratifiedKFold

# X: data (pandas dataframe)
# y: labels (pandas series)

# Create a cross-validation
skf = StratifiedKFold(n_splits=3)

# Split data in training and test datasets
for train_index, test_index in skf.split(X, y):

    X_train = X.iloc[train_index]
    y_train = y.iloc[train_index]

    X_test = X.iloc[test_index]
    y_test = y.iloc[test_index]
```

Problem N2: Data scaling

- Many ML methods may fail (i.e. they don’t converge) if the features have different scales or if they have too small or too big values.
- This situation is common in omics data.
- Data fitting may be inefficient in such situation, or even impossible.

Solution: scale your data properly before training

Problem N2: Data scaling

Train
Train
Test

Scale your data properly before training

1) **In training dataset <u>only</u>**:

- Calculate mean μ
- Calculate standard deviation σ
- Remove μ and divide by σ for each feature

$$scaled\_expression = \frac{expression - \mu}{\sigma}$$

2) **In test dataset**:

- Transform test dataset using μ and σ values obtained in training dataset
- Do not recalculate μ and σ in test dataset because it can significantly shift test values and result in a poor generalization
- Do not use the whole data (train + test) to calculate μ and σ! Test dataset should never be used in learning process.

Data scaling in Python

python™ scikit learn

```python
from sklearn import preprocessing

# Create a scaler
scaler = preprocessing.StandardScaler()

# Calculate mean and standard deviation
# on training dataset only
scaler.fit(X_train)

# Transform training and test datasets
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Problem N3: n << p

- **In omics data**, the number of samples is usually much less than the number of model parameters (for example, number of genes).

  n = 1 640 samples
  p = 21 875 genes

n << p !

- **In theory**

  n > p: The exact solution may not exist. Data can be inconsistent.
  In this case, we still can calculate a robust "best fit" solution .

  n = p: A unique solution exists.

  n < p:
  Ill-posed problem, data are insufficient to properly constrain the model.
  As a consequence, an infinite number of solutions may exist.
  Artefacts (fake solutions) are possible.
  A poor generalization (overfitting) is expected.

Problem N3: n << p

Main problem: data are insufficient to resolve all features

Possible solutions:

Reduce the number of features

- **Selection**: same features, select only a few of them
- **Extraction**: combine several features together to obtain new features
  - Principal components of PCA
  - Latent variables of auto-encoders (neural networks)

Add supplementary *a priori* information to the model

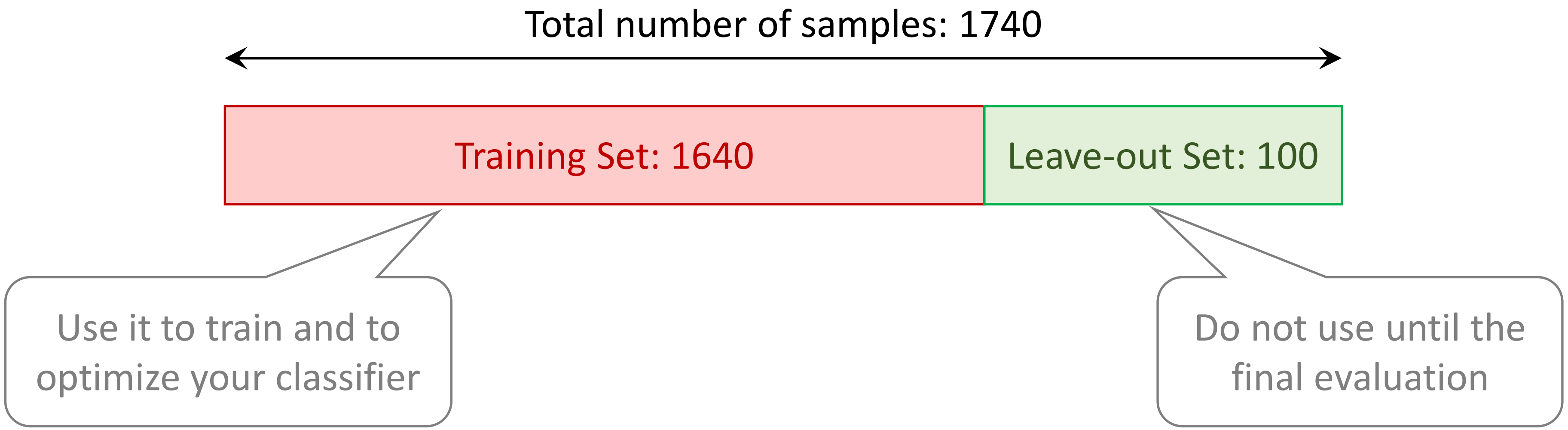
- Add constraints on model behavior (ex. linear model)
- Add constraints on model smoothness (ex. regularization or penalization or dumping)

Prevent "memorization" during learning process

For example, in neural networks, control the learning process using:

- Dropouts
- Early stopping

Use case: leukemia dataset



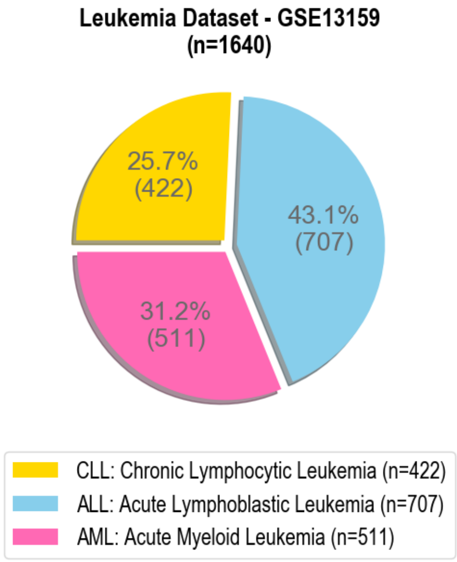
Total number of samples: 1740

Training Set: 1640

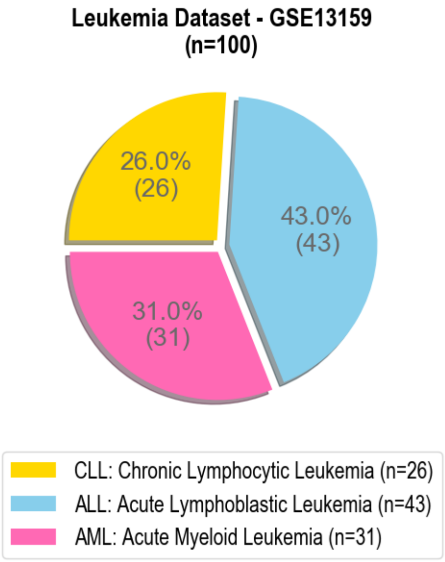
Leave-out Set: 100

Use it to train and to optimize your classifier

Do not use until the final evaluation



Leukemia Dataset - GSE13159 (n=1640)



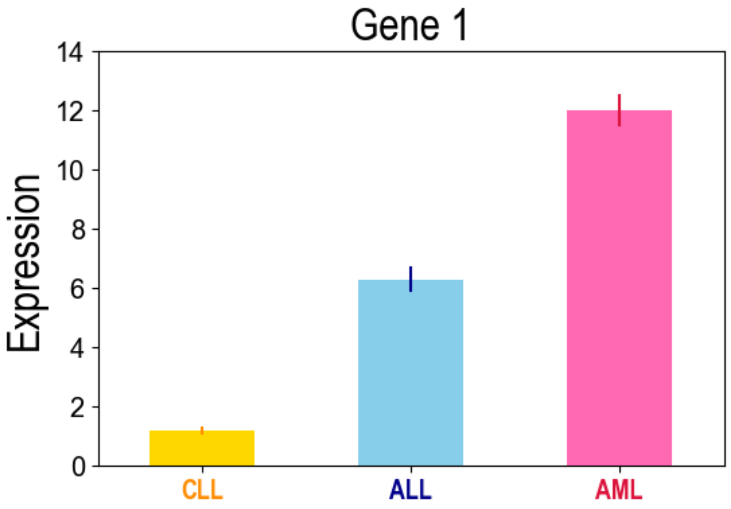
Leukemia Dataset - GSE13159 (n=100)

Variable selection by variance

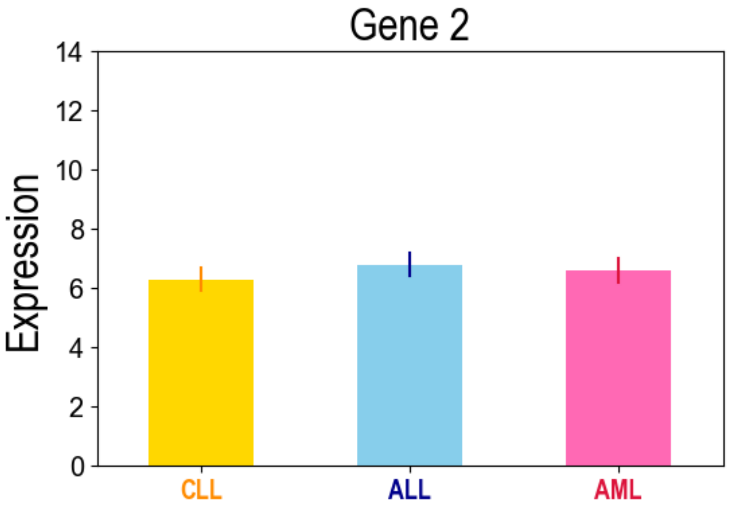
Main idea

Choose genes with a high variance across samples or across groups (CLL, ALL, AML)

If gene expression strongly varies between samples or groups that this gene could probably be a good candidate for prediction.



“Useful” gene?



“Useless” gene?

Variable selection by variance

Main idea

Choose genes with a high variance across samples or across groups (CLL, ALL, AML)

Formal approaches could be:

1) Variance across samples
2) T-Test (for 2 groups) or ANOVA (for more than 2 groups) across groups
3) Expression level greater than a given threshold (above empirically defined noise level, for example, above 1.0 for RPKM values)
4) Output of a linear model

...

Variable selection by variance

Training Set: 1640

Use the training set only

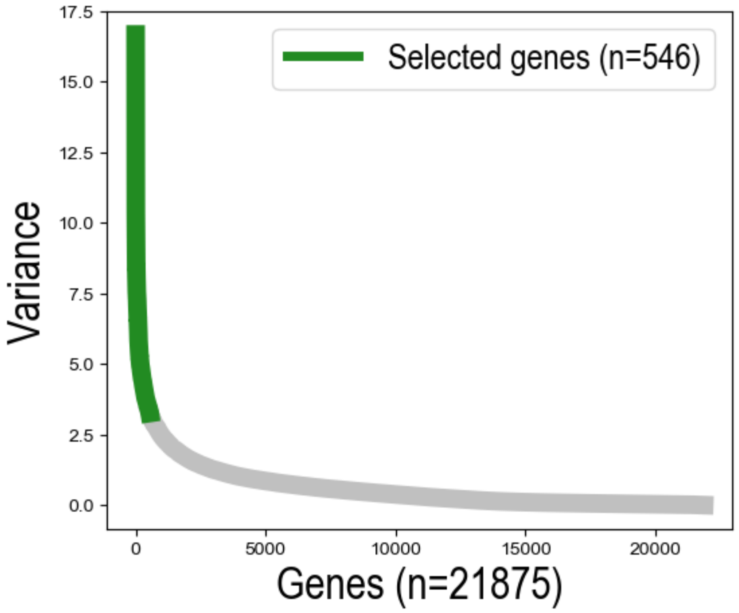
Select only genes with an important variance across samples

How many genes should we select?

- It strongly depends on data and on used model (linear, non-linear, regularization)

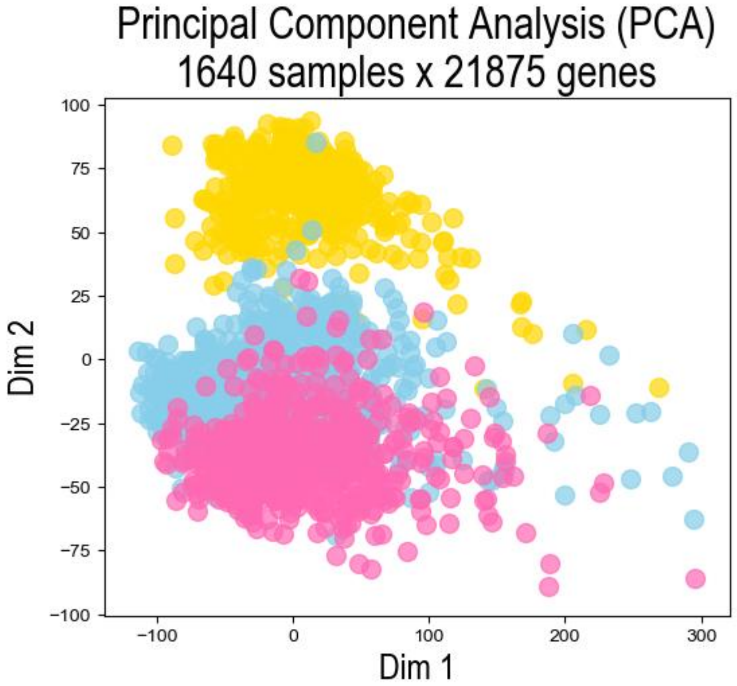
- Sometimes, it is considered that a “good” number of features is approximately equal to 1/3 of the number of samples (empirical guess, similar to use triplica).

Here, 1640 / 3 = 546.

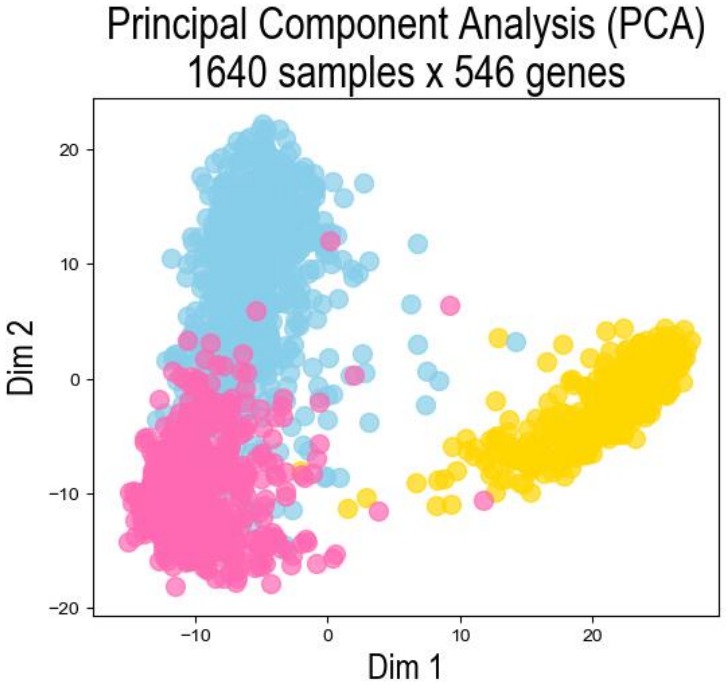


Variable selection by variance

All genes



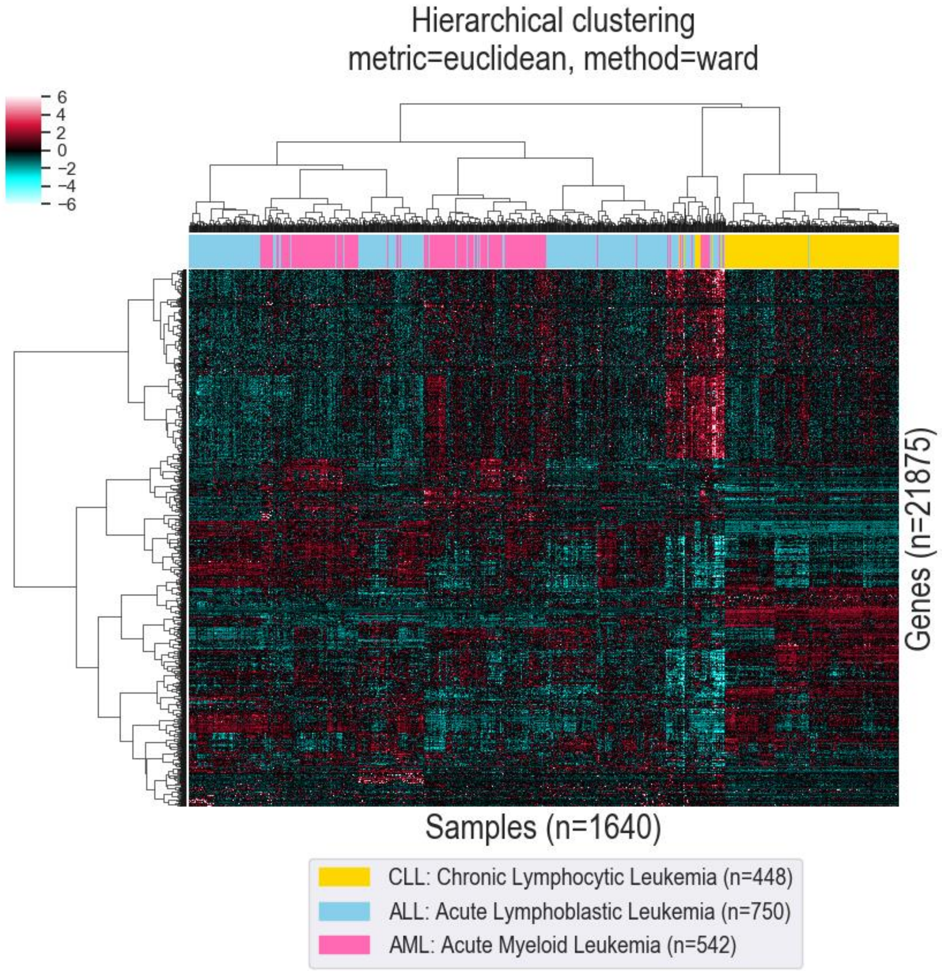
Selected 546 genes



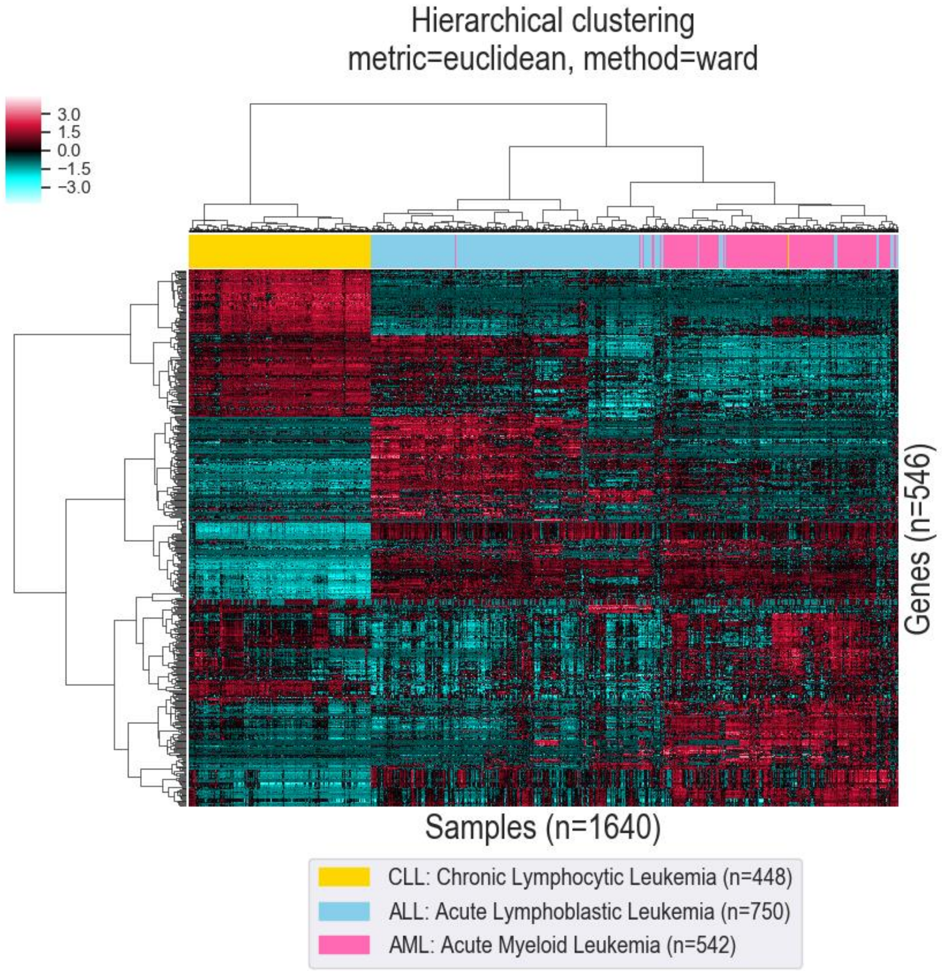
We reduced “noise” and can now see a little better the separation of groups.

Variable selection by variance

All genes



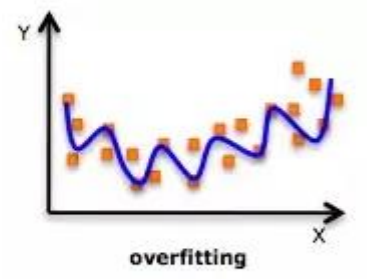
Selected 546 genes



Review

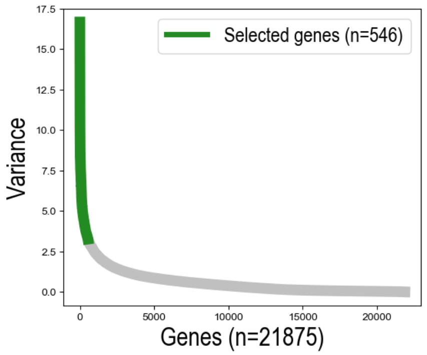
Robustness and generalization

- Results obtained in one study may not work in another one
- Underfitting and overfitting
- Cross-validation



Variable selection and data scaling

- Ill-posed problem if n<<p, artefacts and overfitting are possible
- Reduce the number of variables
- Scale the data properly


Selected genes (n=546)
Variance
Genes (n=21875)