CSI5180. Machine Learning for Bioinformatics Applications

Fundamentals of Machine Learning — Feature Engineering and Data Imputation

by

Marcel Turcotte

Version **November 6, 2019**

Preamble

Preamble

Fundamentals of Machine Learning — Feature Engineering and Data Imputation

This lecture is all about the data. How the amount of data might affect the outcome of the project. How to encode the various data types encountered in bioinformatics. How to scale the data. Finally, how to handle situations where some values are missing.

General objective :

- **Describe** the fundamental concepts of machine learning

Learning objectives

- **Describe** the different ways to encode data, distinguishing the case of ordinal and categorical data.
- **Compare** the different ways to scale numerical values.
- **Explain** the approaches to handle missing values.

Reading:

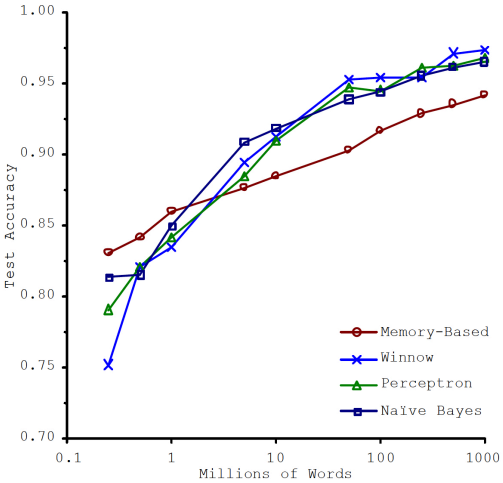
- Ernst, J. & Kellis, M. Large-scale imputation of epigenomic datasets for systematic annotation of diverse human tissues. *Nat Biotechnol* 33, 364376 (2015).
- Durham, T. J., Libbrecht, M. W., Howbert, J. J., Bilmes, J. & Noble, W. S. PREDICTD PaRallel Epigenomics Data Imputation with Cloud-based Tensor Decomposition. *Nature Communications* 9, (2018).

Plan

Data

Size does matter

- **"However, these results suggest that we may want to reconsider the trade-off between spending time and money on algorithm development versus spending it on corpus development algorithms themselves."**
- Banko, M. & Brill, E. Scaling to very very large corpora for natural language disambiguation. Association for Computational Linguistics, 2001.



The Unreasonable Effectiveness of Data

- Halevy, A., Norvig, P. & Pereira, F. The Unreasonable Effectiveness of Data. *IEEE Intelligent Systems* **24**, 812 (2009).
- `https://youtu.be/yvDCzhbjYWs` (01:02:56)

Issues

- "[Y]our training data be **representative of the new cases** you want to generalize to." [1]

Issues

- "[Y]our training data be **representative of the new cases** you want to generalize to." [1]
  - Can a classifier trained on **mouse** data be applied on **human** data?

Issues

- "[Y]our training data be **representative of the new cases** you want to generalize to." [1]
  - Can a classifier trained on **mouse** data be applied on **human** data?
- Ideally, the distribution of the training data and that of the future applications should be similar.

Issues

- "[Y]our training data be **representative of the new cases** you want to generalize to." [1]
  - Can a classifier trained on **mouse** data be applied on **human** data?
- Ideally, the distribution of the training data and that of the future applications should be similar.
  - Learning algorithms learn best on balanced data sets.

Issues

- "[Y]our training data be **representative of the new cases** you want to generalize to." [1]
  - Can a classifier trained on **mouse** data be applied on **human** data?
- Ideally, the distribution of the training data and that of the future applications should be similar.
  - Learning algorithms learn best on balanced data sets.
  - Real-world data might be highly skewed (billions of positions along the genome, few transcriptions start sites).

Issues

- "[Y]our training data be **representative of the new cases** you want to generalize to." [1]
  - Can a classifier trained on **mouse** data be applied on **human** data?
- Ideally, the distribution of the training data and that of the future applications should be similar.
  - Learning algorithms learn best on balanced data sets.
  - Real-world data might be highly skewed (billions of positions along the genome, few transcriptions start sites).
- **Sampling bias**

Issues

- "[Y]our training data be **representative of the new cases** you want to generalize to." [1]
  - Can a classifier trained on **mouse** data be applied on **human** data?
- Ideally, the distribution of the training data and that of the future applications should be similar.
  - Learning algorithms learn best on balanced data sets.
  - Real-world data might be highly skewed (billions of positions along the genome, few transcriptions start sites).
- **Sampling bias**
  - Experiment methods might produce more errors for certain parts of the genome: highly repetitive regions, regions where the DNA is tightly packed.

Issues

- "[Y]our training data be **representative of the new cases** you want to generalize to." [1]
  - Can a classifier trained on **mouse** data be applied on **human** data?
- Ideally, the distribution of the training data and that of the future applications should be similar.
  - Learning algorithms learn best on balanced data sets.
  - Real-world data might be highly skewed (billions of positions along the genome, few transcriptions start sites).
- **Sampling bias**
  - Experiment methods might produce more errors for certain parts of the genome: highly repetitive regions, regions where the DNA is tightly packed.
- **Poor quality**: some experimental methods produce a high number of false positive (protein-protein interactions, ChIP-Seq, etc.).

Issues

- "[Y]our training data be **representative of the new cases** you want to generalize to." [1]
  - Can a classifier trained on **mouse** data be applied on **human** data?
- Ideally, the distribution of the training data and that of the future applications should be similar.
  - Learning algorithms learn best on balanced data sets.
  - Real-world data might be highly skewed (billions of positions along the genome, few transcriptions start sites).
- **Sampling bias**
  - Experiment methods might produce more errors for certain parts of the genome: highly repetitive regions, regions where the DNA is tightly packed.
- **Poor quality**: some experimental methods produce a high number of false positive (protein-protein interactions, ChIP-Seq, etc.).
- A large number of **irrelevant** features might confuse the learning algorithms.

Encoding

Feature Engineering

1. Get **enough** data
2. Extract **features** from the raw data
   - **Labour** intensive
   - Requires **creativity**
   - **Domain knowledge** is a plus!

Feature engineering - sequence information

Some learning algorithms require the data to be represented as **numerical values**.

- **DNA** sequences are made of four letters, **A**, **C**, **G**, **T**.

Feature engineering - sequence information

Some learning algorithms require the data to be represented as **numerical values**.

- **DNA** sequences are made of four letters, **A**, **C**, **G**, **T**.
- Consider the following representations, **which one** do you prefer and **why**?

Feature engineering - sequence information

Some learning algorithms require the data to be represented as **numerical values**.

- **DNA** sequences are made of four letters, **A**, **C**, **G**, **T**.
- Consider the following representations, **which one** do you prefer and **why**?
  1. A = 0, C = 1, G = 2, T = 3

Feature engineering - sequence information

Some learning algorithms require the data to be represented as **numerical values**.

- **DNA** sequences are made of four letters, **A**, **C**, **G**, **T**.
- Consider the following representations, **which one** do you prefer and **why**?
  1. A = 0, C = 1, G = 2, T = 3
  2. A = [0,0], C = [0,1], G = [1,0], T = [1,1]

Feature engineering - sequence information

Some learning algorithms require the data to be represented as **numerical values**.

- **DNA** sequences are made of four letters, **A**, **C**, **G**, **T**.
- Consider the following representations, **which one** do you prefer and **why**?
  1. A = 0, C = 1, G = 2, T = 3
  2. A = [0,0], C = [0,1], G = [1,0], T = [1,1]
  3. A = [1,0,0,0], C = [0,1,0,0], G = [0,0,1,0], T = [0,0,0,1]

Feature engineering - sequence information

Some learning algorithms require the data to be represented as **numerical values**.

- **DNA** sequences are made of four letters, **A**, **C**, **G**, **T**.
- Consider the following representations, **which one** do you prefer and **why**?
  1. A = 0, C = 1, G = 2, T = 3
  2. A = [0,0], C = [0,1], G = [1,0], T = [1,1]
  3. A = [1,0,0,0], C = [0,1,0,0], G = [0,0,1,0], T = [0,0,0,1]
- The latter is called **one-hot encoding** and it should be preferred for **categorical** data.

Feature engineering - sequence information

Some learning algorithms require the data to be represented as **numerical values**.

- **DNA** sequences are made of four letters, **A**, **C**, **G**, **T**.
- Consider the following representations, **which one** do you prefer and **why**?
  1. A = 0, C = 1, G = 2, T = 3
  2. A = [0,0], C = [0,1], G = [1,0], T = [1,1]
  3. A = [1,0,0,0], C = [0,1,0,0], G = [0,0,1,0], T = [0,0,0,1]
- The latter is called **one-hot encoding** and it should be preferred for **categorical** data.
- This increases the dimensionality of the feature vectors.

Feature engineering - sequence information

Some learning algorithms require the data to be represented as **numerical values**.

- **DNA** sequences are made of four letters, **A**, **C**, **G**, **T**.
- Consider the following representations, **which one** do you prefer and **why**?
  1. A = 0, C = 1, G = 2, T = 3
  2. A = [0,0], C = [0,1], G = [1,0], T = [1,1]
  3. A = [1,0,0,0], C = [0,1,0,0], G = [0,0,1,0], T = [0,0,0,1]
- The latter is called **one-hot encoding** and it should be preferred for **categorical** data.
- This increases the dimensionality of the feature vectors.
- The other encodings are introducing a **bias**. With the first two encodings, we are saying that **A** and **C** are somewhat similar, but **A** and **T** are not!

Feature engineering - sequence information

Some learning algorithm requires the data to be represented as **numerical values**.

- **DNA** sequences are made of four letters, **A**, **C**, **G**, **T**.

Feature engineering - sequence information

Some learning algorithm requires the data to be represented as **numerical values**.

- **DNA** sequences are made of four letters, **A**, **C**, **G**, **T**.
- From **assignment 1**, we have seen that sequences can be represented by *l*-**words** (grams) frequency vectors.

Feature engineering - sequence information

Some learning algorithm requires the data to be represented as **numerical values**.

- **DNA** sequences are made of four letters, **A**, **C**, **G**, **T**.
- From **assignment 1**, we have seen that sequences can be represented by *l*-**words** (grams) frequency vectors.
- Compare **one-hot** and *l*-**words**.

Feature engineering - sequence information

Some learning algorithm requires the data to be represented as **numerical values**.

- **DNA** sequences are made of four letters, **A**, **C**, **G**, **T**.
- From **assignment 1**, we have seen that sequences can be represented by *l*-**words** (grams) frequency vectors.
- Compare **one-hot** and *l*-**words**.
- Later in the semester, we will consider an additional encoding called **embedding**.

sklearn.preprocessing.OneHotEncoder

```
from numpy import array
from sklearn.preprocessing import OneHotEncoder

data = ['T','T','C','T','G','G','C','A','C','T','T','G']

values = array(data)
values = values.reshape(len(values),1)

onehot_encoder = OneHotEncoder()
onehot_encoder.fit(values)

values_encoded = onehot_encoder.transform(values)

print(values_encoded.toarray())
```

- **Save the encoding** and use on your validation set, test set, and production!

keras.utils.to_categorical

```
import numpy as np
from sklearn.preprocessing import LabelEncoder
from keras.utils import to_categorical

data = ['T','T','C','T','G','G','C','A','C','T','T','G']

values = array(data)
values = values.reshape(len(values),1)

label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(values)

data_encoded = to_categorical(integer_encoded)

print(data_encoded)
```

pandas.get_dummies

```
import pandas as pd

data = list('TTCTGGCACTTGGTTGTTCT')

onehot_encoded = pd.get_dummies(pd.Series(data))

print(onehot_encoded)
```

```
   A  C  G  T
0  0  0  0  1
1  0  0  0  1
2  0  1  0  0
3  0  0  0  1
4  0  0  1  0
5  0  0  1  0
...
```

Feature engineering - ordinal

- **Categorical** data should not be encoded with ordered numbers.
- **Ordinal** data can be encoded with **ordered numbers**.
  - **Resolution:** Poor = 1, Average = 2, Good = 3, Excellent = 4

Feature engineering - binning

- Occasionally, you might want to regroup ordinal values into **bins** (**buckets**).

Feature engineering - binning

- Occasionally, you might want to regroup ordinal values into **bins** (**buckets**).
- **Start position:**
  1 to 20 = **n-terminal**, -1 to -20 = **c-terminal**, otherwise = **core**.

Feature engineering - binning

- Occasionally, you might want to regroup ordinal values into **bins** (**buckets**).
- **Start position:**
  1 to 20 = **n-terminal**, -1 to -20 = **c-terminal**, otherwise = **core**.
- **Pros:** might allow the algorithm to learn using less training examples.

Feature engineering - binning

- Occasionally, you might want to regroup ordinal values into **bins** (**buckets**).
- **Start position:**
  1 to 20 = **n-terminal**, -1 to -20 = **c-terminal**, otherwise = **core**.
- **Pros:** might allow the algorithm to learn using less training examples.
- **Cons:** could require domain expertise to create meaningful categories might fail to generalize, perhaps **n-terminal** should have been defined as 1 to 21, 1 to 22, etc.

Scaling

Normalization

- Many learning algorithms work best if the numerical values of the features have **similar range of values**, say [-1,1] or [0,1].
  - Namely, the optimization (say gradient descent) may converge more rapidly.

Normalization:

$$\frac{x_i^{(j)} - \min^{(j)}}{\max^{(j)} - \min^{(j)}}$$

See: sklearn.preprocessing.MinMaxScaler

Standardization (or z-score normalization)

With **standardization**, each feature has a **normal distribution**, with $\mu = 0$ and $\sigma = 1$.

$$\frac{x_i^{(j)} - \mu^{(j)}}{\sigma^{(j)}}$$

- Note that the range of values is not bounded!

See: sklearn.preprocessing.StandardScaler

Normalization or standardization?

Normalization or standardization?

- Treat scaling as a hyperparameter, **try both**, normalization and standardization.

Normalization or standardization?

- Treat scaling as a hyperparameter, **try both**, normalization and standardization.
- **Standardization** should be less affected by **outliers** than **normalization**. Do you see why?

Normalization or standardization?

- Treat scaling as a hyperparameter, **try both**, normalization and standardization.
- **Standardization** should be less affected by **outliers** than **normalization**. Do you see why?
- According to [2] §5, in general:

Normalization or standardization?

- Treat scaling as a hyperparameter, **try both**, normalization and standardization.
- **Standardization** should be less affected by **outliers** than **normalization**. Do you see why?
- According to [2] §5, in general:
  - Use **standardization** with **unsupervised learning**

Normalization or standardization?

- Treat scaling as a hyperparameter, **try both**, normalization and standardization.
- **Standardization** should be less affected by **outliers** than **normalization**. Do you see why?
- According to [2] §5, in general:
  - Use **standardization** with **unsupervised learning**
  - Use **standardization** if the values of the features are **normally distributed**

Normalization or standardization?

- Treat scaling as a hyperparameter, **try both**, normalization and standardization.
- **Standardization** should be less affected by **outliers** than **normalization**. Do you see why?
- According to [2] §5, in general:
  - Use **standardization** with **unsupervised learning**
  - Use **standardization** if the values of the features are **normally distributed**
  - If there are **outliers** use **standardization**, see above.

Normalization or standardization?

- Treat scaling as a hyperparameter, **try both**, normalization and standardization.
- **Standardization** should be less affected by **outliers** than **normalization**. Do you see why?
- According to [2] §5, in general:
  - Use **standardization** with **unsupervised learning**
  - Use **standardization** if the values of the features are **normally distributed**
  - If there are **outliers** use **standardization**, see above.
  - Else, use **normalization**

Pipeline

Recall

As discussed in **Essential Bioinformatics skills**, write scripts for everything!

- It **documents** your project
- This allows to **redo** the work
- With time, you will be building a reusable library of functions for your specific domain
- As **new data** become available, you will be able to retrain your learning algorithm

sklearn.pipeline.Pipeline

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

num_pipeline = Pipeline([
        ('imputer', SimpleImputer(strategy="median")),
        ('attribs_adder', CombinedAttributesAdder()),
        ('std_scaler', StandardScaler()),
        ])

training_num_tr = num_pipeline.fit_transform(training_num)

full_pipeline = ColumnTransformer([
        ("num", num_pipeline, num_attribs),
        ("cat", OneHotEncoder(), ["sequence"]),
    ])

training_prepared = full_pipeline.fit_transform(training)
```

Missing values

Missing values

- What **can you do** or **should you do** if some values are missing?

Missing values

- What **can you do** or **should you do** if some values are missing?
  - **DNase sensitivity** footprinting data is not available for a particular **cell line**.

Missing values

- What **can you do** or **should you do** if some values are missing?
  - **DNase sensitivity** footprinting data is not available for a particular **cell line**.
- You can **drop examples** for which data is missing.

Missing values

- What **can you do** or **should you do** if some values are missing?
  - **DNase sensitivity** footprinting data is not available for a particular **cell line**.
- You can **drop examples** for which data is missing.
  - Can only be done if the data set is **large enough** and this **will not impact** the outcome of the project.

Missing values

- What **can you do** or **should you do** if some values are missing?
  - **DNase sensitivity** footprinting data is not available for a particular **cell line**.
- You can **drop examples** for which data is missing.
  - Can only be done if the data set is **large enough** and this **will not impact** the outcome of the project.
- You could **drop features** for which data is missing.

Missing values

- What **can you do** or **should you do** if some values are missing?
  - **DNase sensitivity** footprinting data is not available for a particular **cell line**.
- You can **drop examples** for which data is missing.
  - Can only be done if the data set is **large enough** and this **will not impact** the outcome of the project.
- You could **drop features** for which data is missing.
  - Can only be done if this **will not impact** the outcome of the project.

Missing values

- What **can you do** or **should you do** if some values are missing?
  - **DNase sensitivity** footprinting data is not available for a particular **cell line**.
- You can **drop examples** for which data is missing.
  - Can only be done if the data set is **large enough** and this **will not impact** the outcome of the project.
- You could **drop features** for which data is missing.
  - Can only be done if this **will not impact** the outcome of the project.
- Use a **learning algorithm** that handles missing data, say XGBoost.

Missing values

- What **can you do** or **should you do** if some values are missing?
  - **DNase sensitivity** footprinting data is not available for a particular **cell line**.
- You can **drop examples** for which data is missing.
  - Can only be done if the data set is **large enough** and this **will not impact** the outcome of the project.
- You could **drop features** for which data is missing.
  - Can only be done if this **will not impact** the outcome of the project.
- Use a **learning algorithm** that handles missing data, say XGBoost.
  - Not always feasible, sklearn.linear_model.LinearRegression would throw an exception.

Missing values

- What **can you do** or **should you do** if some values are missing?
  - **DNase sensitivity** footprinting data is not available for a particular **cell line**.
- You can **drop examples** for which data is missing.
  - Can only be done if the data set is **large enough** and this **will not impact** the outcome of the project.
- You could **drop features** for which data is missing.
  - Can only be done if this **will not impact** the outcome of the project.
- Use a **learning algorithm** that handles missing data, say XGBoost.
  - Not always feasible, sklearn.linear_model.LinearRegression would throw an exception.
- **Data imputation**

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.
  - Replace the missing values by the **mean** or **median** value for that **feature** (column).

$$\frac{1}{N}\sum_{i=1}^{N} x_i^{(j)}$$

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy="median")

X = imputer.transform(training_num)
```

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.
  - Replace the missing values by the **mean** or **median** value for that **feature** (column).

$$\frac{1}{N}\sum_{i=1}^{N} x_i^{(j)}$$

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy="median")

X = imputer.transform(training_num)
```

  - **Cons:** ignores correlations (complex relationships) between features!

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.
  - Replace the missing values by the **mean** or **median** value for that **feature** (column).

$$\frac{1}{N}\sum_{i=1}^{N} x_i^{(j)}$$

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy="median")

X = imputer.transform(training_num)
```

    - **Cons:** ignores correlations (complex relationships) between features!
  - A related technique consists of replacing the missing values by the **most frequent value** for that feature, with the same drawback as above.

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.
  - Replace the missing values with a **value outside of the normal range**, assuming that your data has been normalized, range [0,1], use -1 or 2 as a special value.

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.
  - Replace the missing values with a **value outside of the normal range**, assuming that your data has been normalized, range [0,1], use -1 or 2 as a special value.
    - Here, you are hoping that the **learning algorithm** will **learn how to handle** missing values.

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.
  - Replace the missing values with a **value in the middle of the normal range**, assuming that your data is distributed in the range [-1,1], use 0.

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.
  - Replace the missing values with a **value in the middle of the normal range**, assuming that your data is distributed in the range [-1,1], use 0.
  - For categorical data, use small non-zero numerical values.

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.
  - Replace the missing values with a **value in the middle of the normal range**, assuming that your data is distributed in the range [-1,1], use 0.
  - For categorical data, use small non-zero numerical values.
    - Say [0.25, 0.25, 0.25, 0.25] where you would have used [1, 0, 0, 0] for A, [0, 1, 0, 0] for C, [0, 0, 1, 0] for G, [0, 0, 0, 1] for T.

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.
  - Replace the missing values with a **value in the middle of the normal range**, assuming that your data is distributed in the range [-1,1], use 0.
  - For categorical data, use small non-zero numerical values.
    - Say [0.25, 0.25, 0.25, 0.25] where you would have used [1, 0, 0, 0] for A, [0, 1, 0, 0] for C, [0, 0, 1, 0] for G, [0, 0, 0, 1] for T.
  - The **hope** is that those intermediate values will not affect too negatively the results.

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.
  - Replace the missing values with a **value in the middle of the normal range**, assuming that your data is distributed in the range [-1,1], use 0.
  - For categorical data, use small non-zero numerical values.
    - Say [0.25, 0.25, 0.25, 0.25] where you would have used [1, 0, 0, 0] for A, [0, 1, 0, 0] for C, [0, 0, 1, 0] for G, [0, 0, 0, 1] for T.
  - The **hope** is that those intermediate values will not affect too negatively the results.
- In all the cases, you **cannot know in advance** which method works best, you will have compare several methods and use the one that works best.

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.
  - **What else** can you do?

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.
  - **What else** can you do?
  - What is the **problem** that we are trying to solve?

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.
  - **What else** can you do?
  - What is the **problem** that we are trying to solve?
    - **Predict** some unknown label for a given example.

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.
  - **What else** can you do?
  - What is the **problem** that we are trying to solve?
    - **Predict** some unknown label for a given example.
    - Does this sound **familiar** to you?

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.
  - **What else** can you do?
  - What is the **problem** that we are trying to solve?
    - **Predict** some unknown label for a given example.
    - Does this sound **familiar** to you?
    - Indeed, this can be seen as a **supervised learning** problem.

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.
  - **What else** can you do?
  - What is the **problem** that we are trying to solve?
    - **Predict** some unknown label for a given example.
    - Does this sound **familiar** to you?
    - Indeed, this can be seen as a **supervised learning** problem.
    - Let $\hat{x}_i$ be a new example, $[x_i^{(1)}, x_i^{(2)}, x_i^{(j-1)}, x_i^{(j+1)}, \ldots, x_i^{(D)}]$ and $\hat{y}_i = x_i^j$.

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.
  - **What else** can you do?
  - What is the **problem** that we are trying to solve?
    - **Predict** some unknown label for a given example.
    - Does this sound **familiar** to you?
    - Indeed, this can be seen as a **supervised learning** problem.
    - Let $\hat{x}_i$ be a new example, $[x_i^{(1)}, x_i^{(2)}, x_i^{(j-1)}, x_i^{(j+1)}, \ldots, x_i^{(D)}]$ and $\hat{y}_i = x_i^j$.
    - Use all the examples x_i for which x_i^j is **not** missing as **training set**.

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.
  - **What else** can you do?
  - What is the **problem** that we are trying to solve?
    - **Predict** some unknown label for a given example.
    - Does this sound **familiar** to you?
    - Indeed, this can be seen as a **supervised learning** problem.
    - Let $\hat{x}_i$ be a new example, $[x_i^{(1)}, x_i^{(2)}, x_i^{(j-1)}, x_i^{(j+1)}, \ldots, x_i^{(D)}]$ and $\hat{y}_i = x_i^j$.
    - Use all the examples x_i for which x_i^j is **not** missing as **training set**.
    - Train a classifier, which you will use the predict (impute) the missing values.

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.
  - Using a **learning algorithm**:

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.
  - Using a **learning algorithm**:
    1. Use an **instance-based** method such k **nearest neighbours** (k-NN) to find the k closest examples. Use the non-missing value from the neighbourhood.

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.
  - Using a **learning algorithm**:
    1. Use an **instance-based** method such k **nearest neighbours** (k-NN) to find the k closest examples. Use the non-missing value from the neighbourhood.
    2. Use a **model-based** method such as **random forest**, **tensor decomposition**, or **deep neural networks**.

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.
  - Using a **learning algorithm**:
    1. Use an **instance-based** method such k **nearest neighbours** (k-NN) to find the k closest examples. Use the non-missing value from the neighbourhood.
    2. Use a **model-based** method such as **random forest**, **tensor decomposition**, or **deep neural networks**.
  - **Why?**

Data imputation

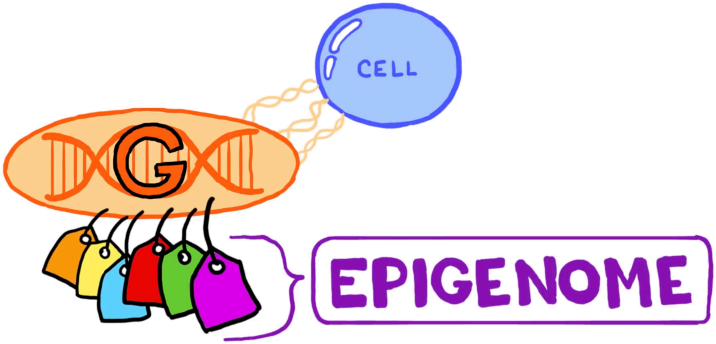
- **Data imputation** is the process of replacing the missing values by computed values.
  - Using a **learning algorithm**:
    1. Use an **instance-based** method such *k* **nearest neighbours** (k-NN) to find the *k* closest examples. Use the non-missing value from the neighbourhood.
    2. Use a **model-based** method such as **random forest**, **tensor decomposition**, or **deep neural networks**.
  - **Why?**
    - These approaches can potentially **handle complex relationships (correlations) between features**!

Data imputation

- **Data imputation** is the process of replacing the missing values by computed values.
  - Using a **learning algorithm**:
    1. Use an **instance-based** method such *k* **nearest neighbours** (k-NN) to find the *k* closest examples. Use the non-missing value from the neighbourhood.
    2. Use a **model-based** method such as **random forest**, **tensor decomposition**, or **deep neural networks**.
  - **Why?**
    - These approaches can potentially **handle complex relationships (correlations) between features**!
    - **However**, these approaches are cost intensive (labour, CPU time, memory, etc.).

Case study

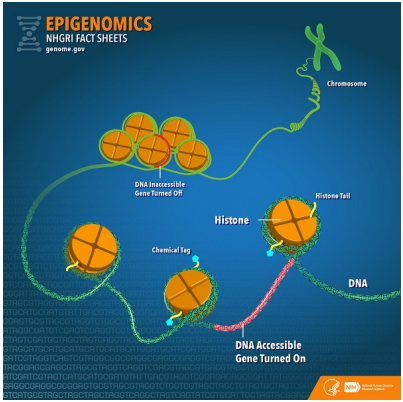
What is epigenetics?



https://youtu.be/_aAhcNjmvhc

What is epigenetics?

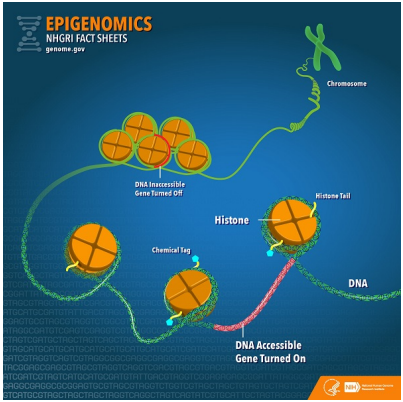
- "Chemical compounds that are **added to single genes** can regulate their activity; these modifications are known as epigenetic changes."



Source: `https://ghr.nlm.nih.gov/primer/howgeneswork/epigenome`

What is epigenetics?

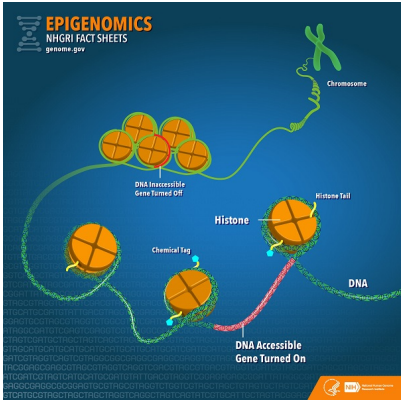
- "Chemical compounds that are **added to single genes** can regulate their activity; these modifications are known as epigenetic changes."
- "The **epigenome** comprises all of the chemical compounds that have been added to the entirety of ones DNA (genome) as a way to regulate the activity (expression) of all the genes within the genome."



Source: `https://ghr.nlm.nih.gov/primer/howgeneswork/epigenome`

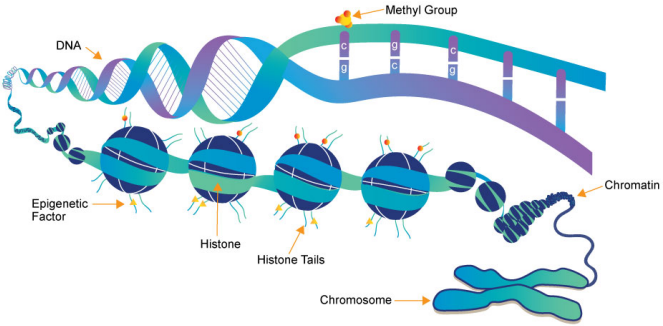
What is epigenetics?

- "Chemical compounds that are **added to single genes** can regulate their activity; these modifications are known as epigenetic changes."
- "The **epigenome** comprises all of the chemical compounds that have been added to the entirety of ones DNA (genome) as a way to regulate the activity (expression) of all the genes within the genome."
- "The chemical compounds of the epigenome are **not** part of the DNA sequence, but are on or attached to DNA (epi- means above in Greek)."



Source: https://ghr.nlm.nih.gov/primer/howgeneswork/epigenome

What is epigenetics?

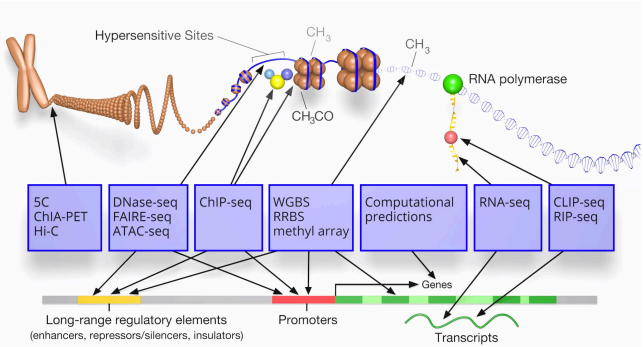


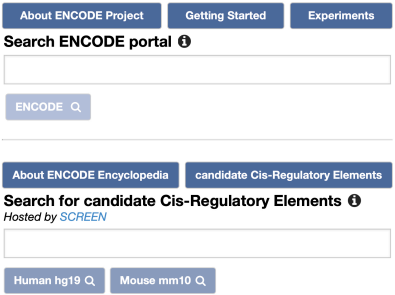
"(...) a change in **phenotype** without a change in **genotype** (...)"

https://www.whatisepigenetics.com/fundamentals/

ENCODE

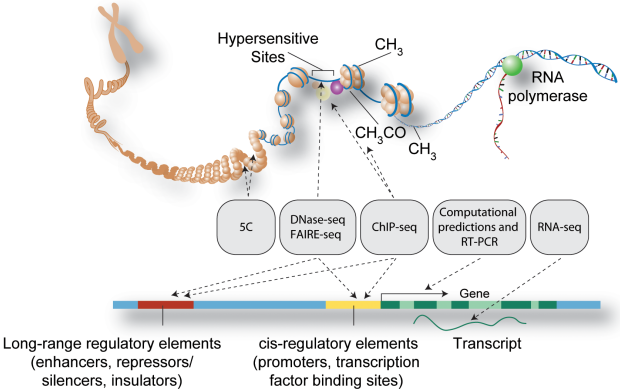
ENCODE: Encyclopedia of DNA Elements





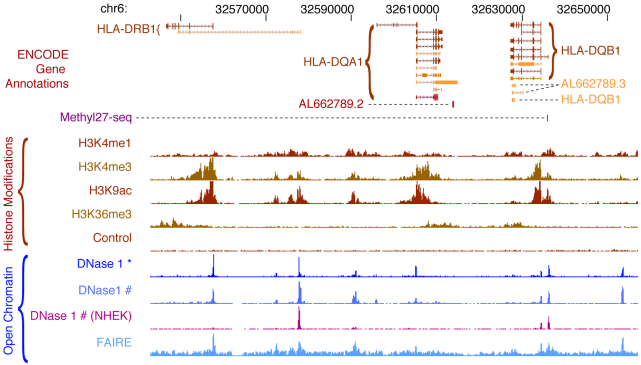
https://www.encodeproject.org

ENCODE - Assays



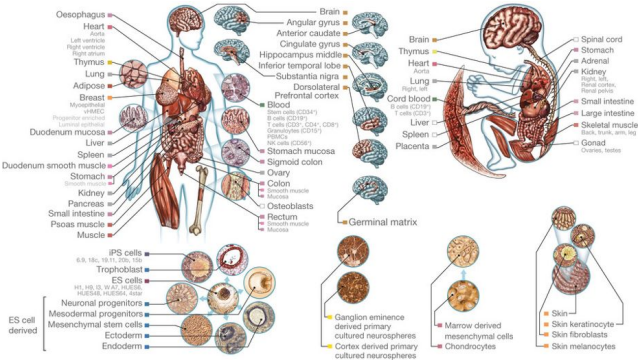
https://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.1001046

ENCODE - Conceptually



https://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.1001046

ENCODE - Cell types/lines

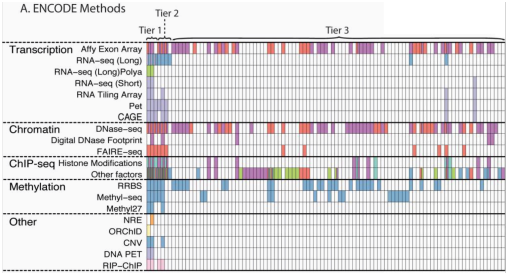


https://www.nature.com/articles/nature14248

ENCODE - Matrix

From W. S. Noble' talk:

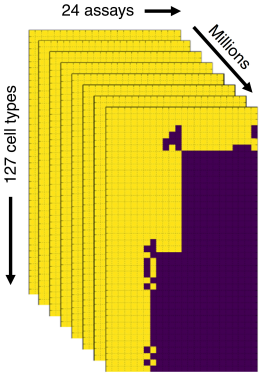
- 359 assay types
- 583 cell types
- Theoretically: 209,297 pairs!
- 5,707 experiments have been done
- The matrix is less than 5 % complete



https://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.1001046

ENCODE - 3D Matrix

- For every assay (359)
  - For every cell line (583)
    - For every position (3.2 Gbp)

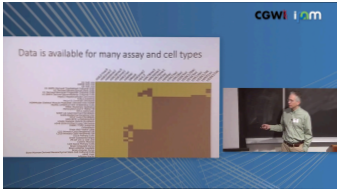


- Durham, T. J., Libbrecht, M. W., Howbert, J. J., Bilmes, J. & Noble, W. S. PREDICTD PaRallel Epigenomics Data Imputation with Cloud-based Tensor Decomposition. *Nature Communications* 9, (2018).

ENCODE - Data imputation

- Ernst, J. & Kellis, M. Large-scale imputation of epigenomic datasets for systematic annotation of diverse human tissues. *Nat Biotechnol* 33, 364376 (2015).
- Durham, T. J., Libbrecht, M. W., Howbert, J. J., Bilmes, J. & Noble, W. S. PREDICTD PaRallel Epigenomics Data Imputation with Cloud-based Tensor Decomposition. *Nature Communications* 9, (2018).
- Schreiber, J. Durham, T., Bilmes, J., and Noble, W. S. Multi-scale deep tensor factorization learns a latent representation of the human epigenome. *bioRxiv*, 2018.

William Noble - IPAM 2018



https://youtu.be/JzSf5AU9VVc (46:13)

- Durham, T. J., Libbrecht, M. W., Howbert, J. J., Bilmes, J. & Noble, W. S. PREDICTD PaRallel Epigenomics Data Imputation with Cloud-based Tensor Decomposition. *Nature Communications* 9, (2018).
- Schreiber, J. Durham, T., Bilmes, J., and Noble, W. S. Multi-scale deep tensor factorization learns a latent representation of the human epigenome. *bioRxiv*, 2018.

Prologue

Summary

- The **amount of data** might be more important than the learning algorithm itself.

Summary

- The **amount of data** might be more important than the learning algorithm itself.
- **Categorical data** can be encoded using the **one-hot encoding**.

Summary

- The **amount of data** might be more important than the learning algorithm itself.
- **Categorical data** can be encoded using the **one-hot encoding**.
- Consider **scaling** the data

Summary

- The **amount of data** might be more important than the learning algorithm itself.
- **Categorical data** can be encoded using the **one-hot encoding**.
- Consider **scaling** the data
- Write **scripts**

Summary

- The **amount of data** might be more important than the learning algorithm itself.
- **Categorical data** can be encoded using the **one-hot encoding**.
- Consider **scaling** the data
- Write **scripts**
- We have seen several approaches for handling **missing values**

Next module

- **Dimensionality reduction**, **feature selection** and **unsupervised learning**.

References

Aurélien Géron.
Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow.
O'Reilly Media, 2nd edition, 2019.

Andriy Burkov.
The Hundred-Page Machine Learning Book.
Andriy Burkov, 2019.

Xuesi Dong, Lijuan Lin, Ruyang Zhang, Yang Zhao, David C Christiani, Yongyue Wei, and Feng Chen.
TOBMI: trans-omics block missing data imputation using a k-nearest neighbor weighted approach.
Bioinformatics, 35(8):1278–1283, Apr 2019.

Kohbalan Moorthy, Aws Naser Jaber, Mohd Arfian Ismail, Ferda Ernawan, Mohd Saberi Mohamad, and Safaai Deris.
Missing-values imputation algorithms for microarray gene expression data.
Methods Mol Biol, 1986:255–266, 2019.

References

Aiguo Wang, Ye Chen, Ning An, Jing Yang, Lian Li, and Lili Jiang.
Microarray missing value imputation: A regularized local learning method.
IEEE/ACM Trans Comput Biol Bioinform, Feb 2018.

Kohbalan Moorthy, Mohd Saberi Mohamad, and Safaai Deris.
A review on missing value imputation algorithms for microarray gene expression data.
Current Bioinformatics, 9(1):18–22, 2014.

Qian Qin and Jianxing Feng.
Imputation for transcription factor binding predictions based on deep learning.
PLoS Comput Biol, 13(2):e1005403, 02 2017.

Timothy J Durham, Maxwell W Libbrecht, J Jeffry Howbert, Jeff Bilmes, and William Stafford Noble.
PREDICTD PaRallel epigenomics data imputation with cloud-based tensor decomposition.
Nat Commun, 9(1):1402, 04 2018.

References

Jason Ernst and Manolis Kellis.
Large-scale imputation of epigenomic datasets for systematic annotation of diverse human tissues.
Nat Biotechnol, 33(4):364–76, Apr 2015.

Jacob Schreiber, Timothy Durham, Jeffrey Bilmes, and William Stafford Noble.
Multi-scale deep tensor factorization learns a latent representation of the human epigenome.
bioRxiv, 2018.

Marcel **Turcotte**

Marcel.Turcotte@uOttawa.ca

School of Electrical Engineering and **Computer Science** (EE**CS**)
University of Ottawa