# MACHINE LEARNING IN SYSTEMS BIOLOGY I

Jonas Pleyer

29.06.2022

Freiburg Center for Data Analysis and Modeling (FDM)

# INTRODUCTION

'A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E'
- Tom M. Mitchell [Mit97]

1943  First publication of neural network [MP43]

1943 First publication of neural network [MP43]
1956 Dartmouth Summer Research Project (Birthplace of modern
Machine Learning)

1943 First publication of neural network [MP43]

1956 Dartmouth Summer Research Project (Birthplace of modern Machine Learning)

1965 Nilson Machine Learning for pattern classification [Nil65]

1943 First publication of neural network [MP43]

1956 Dartmouth Summer Research Project (Birthplace of modern Machine Learning)

1965 Nilson Machine Learning for pattern classification [Nil65]

1966 Following years: Many setbacks in Artificial Intelligence called 'AI-Winters'

1943  First publication of neural network [MP43]

1956  Dartmouth Summer Research Project (Birthplace of modern Machine Learning)

1965  Nilson Machine Learning for pattern classification [Nil65]

1966  Following years: Many setbacks in Artificial Intelligence called 'AI-Winters'

1995  Support Vector Machines are first introduced

1943 First publication of neural network [MP43]

1956 Dartmouth Summer Research Project (Birthplace of modern Machine Learning)

1965 Nilson Machine Learning for pattern classification [Nil65]

1966 Following years: Many setbacks in Artificial Intelligence called 'AI-Winters'

1995 Support Vector Machines are first introduced

2002 Torch first release (open source library)

1943 First publication of neural network [MP43]

1956 Dartmouth Summer Research Project (Birthplace of modern Machine Learning)

1965 Nilson Machine Learning for pattern classification [Nil65]

1966 Following years: Many setbacks in Artificial Intelligence called 'AI-Winters'

1995 Support Vector Machines are first introduced

2002 Torch first release (open source library)

2006 Geoffrey Hinton coins 'Deep Learning' [HOT06]

1943 First publication of neural network [MP43]

1956 Dartmouth Summer Research Project (Birthplace of modern Machine Learning)

1965 Nilson Machine Learning for pattern classification [Nil65]

1966 Following years: Many setbacks in Artificial Intelligence called 'AI-Winters'

1995 Support Vector Machines are first introduced

2002 Torch first release (open source library)

2006 Geoffrey Hinton coins 'Deep Learning' [HOT06]

>2006 Companies such as Netflix, Facebook, Microsoft, Google fund projects/prizes in and use machine learning/artificial intelligence

Machine learning techniques follow a similar workflow.

1. Define Problem (scope, feasability)

Machine learning techniques follow a similar workflow.

1. Define Problem (scope, feasability)
2. Gather Data (assumptions, constraints)

Machine learning techniques follow a similar workflow.

1. Define Problem (scope, feasability)
2. Gather Data (assumptions, constraints)
3. Pre-process Data (cleanup, drop)

Machine learning techniques follow a similar workflow.

1. Define Problem (scope, feasability)
2. Gather Data (assumptions, constraints)
3. Pre-process Data (cleanup, drop)
4. Analyze Data (define features, find correlations)

Machine learning techniques follow a similar workflow.

1. Define Problem (scope, feasability)
2. Gather Data (assumptions, constraints)
3. Pre-process Data (cleanup, drop)
4. Analyze Data (define features, find correlations)
5. Prepare Data (transform, normalize, drop)

Machine learning techniques follow a similar workflow.

1. Define Problem (scope, feasability)
2. Gather Data (assumptions, constraints)
3. Pre-process Data (cleanup, drop)
4. Analyze Data (define features, find correlations)
5. Prepare Data (transform, normalize, drop)
6. Evaluate Models (train/test, classify/regress)

Machine learning techniques follow a similar workflow.

1. Define Problem (scope, feasability)
2. Gather Data (assumptions, constraints)
3. Pre-process Data (cleanup, drop)
4. Analyze Data (define features, find correlations)
5. Prepare Data (transform, normalize, drop)
6. Evaluate Models (train/test, classify/regress)
7. Tune Model (cross validation, fine tune parameters)

Machine learning techniques follow a similar workflow.

1. Define Problem (scope, feasability)
2. Gather Data (assumptions, constraints)
3. Pre-process Data (cleanup, drop)
4. Analyze Data (define features, find correlations)
5. Prepare Data (transform, normalize, drop)
6. Evaluate Models (train/test, classify/regress)
7. Tune Model (cross validation, fine tune parameters)
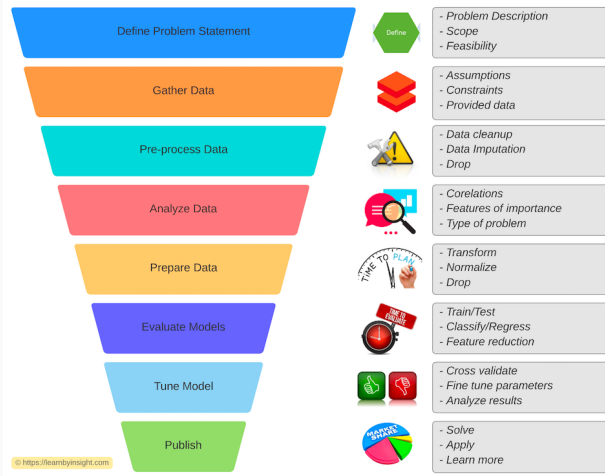8. Apply model to problems, learn more

**Figure 1:** Machine Learning workflow [Mew20]

# CONCEPTS

Supervised

Supervised

► Fit model to labelled data (ie. with 'ground truth')

Supervised

▶ Fit model to labelled data (ie. with 'ground truth')
▶ Data is usually obtained experimentally or assigned by humans

Supervised

- ▶ Fit model to labelled data (ie. with 'ground truth')
- ▶ Data is usually obtained experimentally or assigned by humans
- ▶ Previously labelled data can serve as testing set

Supervised

► Fit model to labelled data (ie. with 'ground truth')
► Data is usually obtained experimentally or assigned by humans
► Previously labelled data can serve as testing set

Supervised

▶ Fit model to labelled data (ie. with 'ground truth')
▶ Data is usually obtained experimentally or assigned by humans
▶ Previously labelled data can serve as testing set

Unsupvervised

Supervised

► Fit model to labelled data (ie. with 'ground truth')
► Data is usually obtained experimentally or assigned by humans
► Previously labelled data can serve as testing set

Unsupvervised

► Data does not contain any labels (only inputs)

Supervised

- ► Fit model to labelled data (ie. with 'ground truth')
- ► Data is usually obtained experimentally or assigned by humans
- ► Previously labelled data can serve as testing set

Unsupvervised

- ► Data does not contain any labels (only inputs)
- ► Find structure in data (clustering, grouping)

Supervised

- ▶ Fit model to labelled data (ie. with 'ground truth')
- ▶ Data is usually obtained experimentally or assigned by humans
- ▶ Previously labelled data can serve as testing set

Unsupvervised

- ▶ Data does not contain any labels (only inputs)
- ▶ Find structure in data (clustering, grouping)

Supervised

▶ Fit model to labelled data (ie. with 'ground truth')
▶ Data is usually obtained experimentally or assigned by humans
▶ Previously labelled data can serve as testing set

Unsupvervised

▶ Data does not contain any labels (only inputs)
▶ Find structure in data (clustering, grouping)

Semi-supervised

Supervised

- ► Fit model to labelled data (ie. with 'ground truth')
- ► Data is usually obtained experimentally or assigned by humans
- ► Previously labelled data can serve as testing set

Unsupvervised

- ► Data does not contain any labels (only inputs)
- ► Find structure in data (clustering, grouping)

Semi-supervised

- ► Combine partly labeled data with partly unlabeled data

This section follows [GKMJ21].

Supervised

- ▶ Fit model to labelled data (ie. with 'ground truth')
- ▶ Data is usually obtained experimentally or assigned by humans
- ▶ Previously labelled data can serve as testing set

Unsupvervised

- ▶ Data does not contain any labels (only inputs)
- ▶ Find structure in data (clustering, grouping)

Semi-supervised

- ▶ Combine partly labeled data with partly unlabeled data
- ▶ Can have huge performance benefits compared to unsupervised learning

This section follows [GKMJ21].

► Classification: Assign datapoints discrete categories (eg. cancerous, non-cancerous). Algorithms are called 'classifiers'.

▶ Classification: Assign datapoints discrete categories (eg. cancerous, non-cancerous). Algorithms are called 'classifiers'.

  If discrete categories are mutually exclusive, we call them 'classes', otherwise 'labels'.

- ▶ Classification: Assign datapoints discrete categories (eg. cancerous, non-cancerous). Algorithms are called 'classifiers'.

  If discrete categories are mutually exclusive, we call them 'classes', otherwise 'labels'.
- ▶ Regression: Output continuous values (eg. predict free energy of protein system).

▶ Classification: Assign datapoints discrete categories (eg. cancerous, non-cancerous). Algorithms are called 'classifiers'.

　If discrete categories are mutually exclusive, we call them 'classes', otherwise 'labels'.

▶ Regression: Output continuous values (eg. predict free energy of protein system).

▶ Classification problems can also be solved with regression and thresholds/binning.

- ▶ Classification: Assign datapoints discrete categories (eg. cancerous, non-cancerous). Algorithms are called 'classifiers'.

  If discrete categories are mutually exclusive, we call them 'classes', otherwise 'labels'.

- ▶ Regression: Output continuous values (eg. predict free energy of protein system).

- ▶ Classification problems can also be solved with regression and thresholds/binning.

- ▶ Clustering: Predict groupings of similar datapoints.

▶ Loss or Cost function:
  Measure deviation to ground truth in supervised learning.
  Implemented similarly in unsupervised situations.

- ▶ Loss or Cost function:
  Measure deviation to ground truth in supervised learning.
  Implemented similarly in unsupervised situations.
- ▶ Parameters: Part of the model, will be adjusted by learning process of the model.

- ▶ Loss or Cost function:
  Measure deviation to ground truth in supervised learning.
  Implemented similarly in unsupervised situations.
- ▶ Parameters: Part of the model, will be adjusted by learning process of the model.
- ▶ Hyperparameters: Not part of the model but control learning process (eg. learning rate, number of iterations)

- ▶ Loss or Cost function:
  Measure deviation to ground truth in supervised learning.
  Implemented similarly in unsupervised situations.
- ▶ Parameters: Part of the model, will be adjusted by learning process of the model.
- ▶ Hyperparameters: Not part of the model but control learning process (eg. learning rate, number of iterations)
- ▶ Training: describes process of iterative learning and adjusting the parameters of the model to obtain better performance. Minimize the loss/cost-function.

- ▶ Loss or Cost function:
  Measure deviation to ground truth in supervised learning.
  Implemented similarly in unsupervised situations.
- ▶ Parameters: Part of the model, will be adjusted by learning process of the model.
- ▶ Hyperparameters: Not part of the model but control learning process (eg. learning rate, number of iterations)
- ▶ Training: describes process of iterative learning and adjusting the parameters of the model to obtain better performance. Minimize the loss/cost-function.
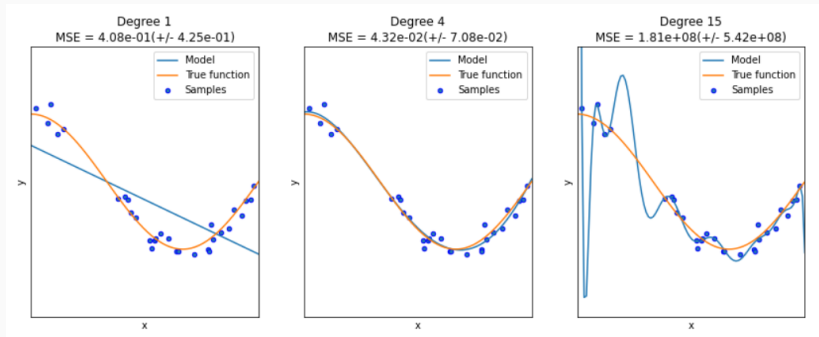- ▶ Validation: Use seperate dataset to test model.

**Figure 2:** Underfitting, Optimal Fitting and Overfitting [Tri20]

▶ Inductive Bias: Set of assumptions.

► Inductive Bias: Set of assumptions.
  Leads it to favour a particular type of solution over others.

► Inductive Bias: Set of assumptions.
Leads it to favour a particular type of solution over others.
Often programmed in mathematical model.

► Inductive Bias: Set of assumptions.
  Leads it to favour a particular type of solution over others.
  Often programmed in mathematical model.
  Example: Recurrent Neural Networks anticipate sequential
  dependencies

- ▶ Inductive Bias: Set of assumptions.
  Leads it to favour a particular type of solution over others.
  Often programmed in mathematical model.
  Example: Recurrent Neural Networks anticipate sequential
  dependencies
- ▶ Trade-off between bias and variance

▶ Inductive Bias: Set of assumptions.
  Leads it to favour a particular type of solution over others.
  Often programmed in mathematical model.
  Example: Recurrent Neural Networks anticipate sequential
  dependencies
▶ Trade-off between bias and variance
  Different inductive biases typically lead to better performance,
  but higher constraints on the model.

▶ Inductive Bias: Set of assumptions.
Leads it to favour a particular type of solution over others.
Often programmed in mathematical model.
Example: Recurrent Neural Networks anticipate sequential
dependencies

▶ Trade-off between bias and variance
Different inductive biases typically lead to better performance,
but higher constraints on the model.
Lower bias makes fewer assumptions.

- ▶ Inductive Bias: Set of assumptions.
  Leads it to favour a particular type of solution over others.
  Often programmed in mathematical model.
  Example: Recurrent Neural Networks anticipate sequential
  dependencies
- ▶ Trade-off between bias and variance
  Different inductive biases typically lead to better performance,
  but higher constraints on the model.
  Lower bias makes fewer assumptions.
- ▶ Variance: How much does trained model change in response to
  training on different dataset.

► Inductive Bias: Set of assumptions.
  Leads it to favour a particular type of solution over others.
  Often programmed in mathematical model.
  Example: Recurrent Neural Networks anticipate sequential
  dependencies
► Trade-off between bias and variance
  Different inductive biases typically lead to better performance,
  but higher constraints on the model.
  Lower bias makes fewer assumptions.
► Variance: How much does trained model change in response to
  training on different dataset.
► We want low bias and low variance.

- ► Inductive Bias: Set of assumptions.
  Leads it to favour a particular type of solution over others.
  Often programmed in mathematical model.
  Example: Recurrent Neural Networks anticipate sequential
  dependencies
- ► Trade-off between bias and variance
  Different inductive biases typically lead to better performance,
  but higher constraints on the model.
  Lower bias makes fewer assumptions.
- ► Variance: How much does trained model change in response to
  training on different dataset.
- ► We want low bias and low variance.
- ► Low bias and low variance often conflict each other.

► Inductive Bias: Set of assumptions.
  Leads it to favour a particular type of solution over others.
  Often programmed in mathematical model.
  Example: Recurrent Neural Networks anticipate sequential
  dependencies
► Trade-off between bias and variance
  Different inductive biases typically lead to better performance,
  but higher constraints on the model.
  Lower bias makes fewer assumptions.
► Variance: How much does trained model change in response to
  training on different dataset.
► We want low bias and low variance.
► Low bias and low variance often conflict each other.
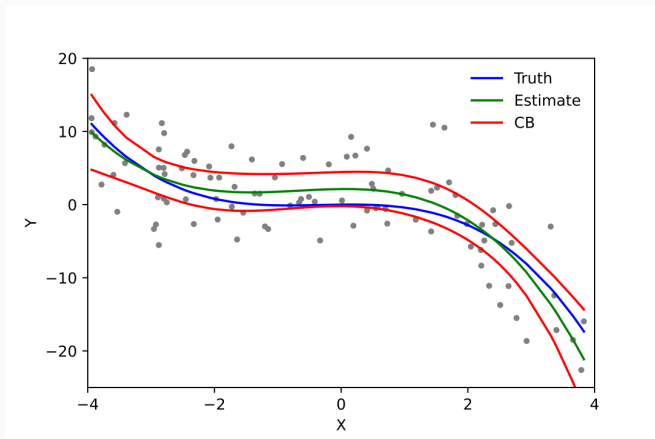  ⇒ Need to balance between them

# MACHINE LEARNING TECHNIQUES
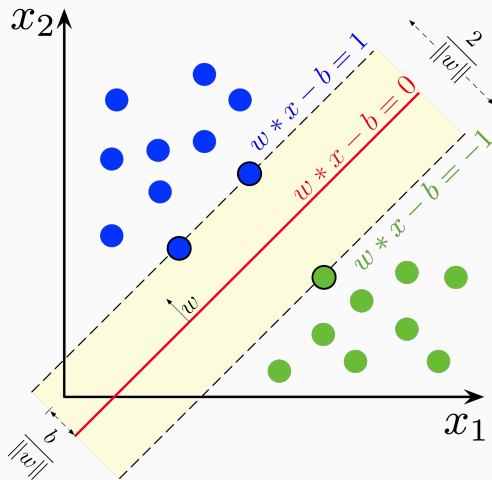
Figure 3: Polynomial Regression [Skb09]

Figure 4: Support Vector Machine [Lar18]

Traditional methods

▶ Linear, polinomial, logistic regression

Traditional methods

- ▶ Linear, polinomial, logistic regression
- ▶ Decision Trees

Traditional methods

- ▶ Linear, polinomial, logistic regression
- ▶ Decision Trees

  Gradient boosting (eg. XGBoost)

Traditional methods

- ▶ Linear, polinomial, logistic regression
- ▶ Decision Trees

  Gradient boosting (eg. XGBoost)
- ▶ Support vector machine

Traditional methods

- ▶ Linear, polinomial, logistic regression
- ▶ Decision Trees
  Gradient boosting (eg. XGBoost)
- ▶ Support vector machine
- ▶ Random Forest

Traditional methods

- ► Linear, polinomial, logistic regression
- ► Decision Trees

  Gradient boosting (eg. XGBoost)
- ► Support vector machine
- ► Random Forest
- ► Genetic Algorithms

- ▶ Traditional machine learning routines should be preferred over neural networks
- ▶ Deep learning can be powerful (and trendy)

  However, currently still limited in applicability

  Requires lots of data $\Rightarrow$ often not present
- ▶ Traditional Methods are faster to develop and test

  They typically expect same number of features with each datapoint

  $\Rightarrow$ Padding and Windowing are methods to circumvent this
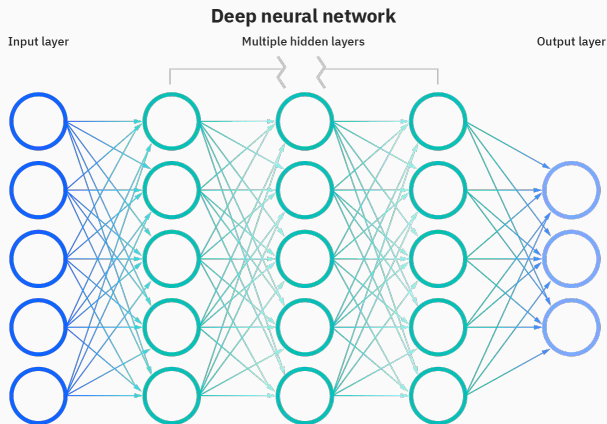
**Figure 5:** Standard, fully connected neural network [Edu20]

- ▶ Universal function approximators
- ▶ No guarantee that model will yield accurate predictins for new data
- ▶ Question: Is the trained model optimal?
- ▶ Neurons are at the heart of Neural Networks

  They apply a function to the input variables $x_i$ to obtain output $y$ by multiplying with learnable weight $w_i$.

$$y = \sigma \left( \sum_{i=1}^{n} w_i x_i + b \right) \tag{1}$$

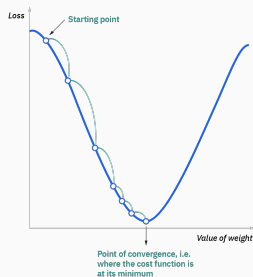- ▶ Multiple layers $\Rightarrow$ iterate this procedure

**Figure 6:** Iterative approach to finding a local optimum of the cost function. It penalizes or rewards good/bad results. An example can be given by the mean squared error [Edu20].

$$MSE = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y} - y)^2 \qquad (2)$$

1. Let $g(x_i)$ be a network output of input variables $x_i$.

1. Let $g(x_i)$ be a network output of input variables $x_i$.
2. Let $C(y_i, g(x_i))$ be the loss function for predicted output $g(x_i)$ and target output $y_i$.

1. Let $g(x_i)$ be a network output of input variables $x_i$.
2. Let $C(y_i, g(x_i))$ be the loss function for predicted output $g(x_i)$ and target output $y_i$.
3. Let $W^l = (w^l_{jk})$ be the weights between layer $l - 1$ and $l$ where $w^l_{jk}$ is the weight between kth node in layer $l - 1$ and jth node in l.

1. Let $g(x_i)$ be a network output of input variables $x_i$.
2. Let $C(y_i, g(x_i))$ be the loss function for predicted output $g(x_i)$ and target output $y_i$.
3. Let $W^l = (w^l_{jk})$ be the weights between layer $l - 1$ and $l$ where $w^l_{jk}$ is the weight between kth node in layer $l - 1$ and jth node in $l$.
4. Calculate the gradient of loss function in weight-space for fixed input-output pair $(x_i, y_i)$

$$\frac{\partial C}{\partial w^l_{jk}}(y_i, g(x_i)) \tag{3}$$

1. Let $g(x_i)$ be a network output of input variables $x_i$.
2. Let $C(y_i, g(x_i))$ be the loss function for predicted output $g(x_i)$ and target output $y_i$.
3. Let $W^l = (w_{jk}^l)$ be the weights between layer $l - 1$ and $l$ where $w_{jk}^l$ is the weight between kth node in layer $l - 1$ and jth node in l.
4. Calculate the gradient of loss function in weight-space for fixed input-output pair $(x_i, y_i)$

$$\frac{\partial C}{\partial w_{jk}^l}(y_i, g(x_i)) \tag{3}$$

5. Multiple layers in Neural Networks mean that $\partial C / \partial w$ needs to be evaluated by chain rule.

1. Let $g(x_i)$ be a network output of input variables $x_i$.
2. Let $C(y_i, g(x_i))$ be the loss function for predicted output $g(x_i)$ and target output $y_i$.
3. Let $W^l = (w_{jk}^l)$ be the weights between layer $l-1$ and $l$ where $w_{jk}^l$ is the weight between kth node in layer $l-1$ and jth node in l.
4. Calculate the gradient of loss function in weight-space for fixed input-output pair $(x_i, y_i)$

$$\frac{\partial C}{\partial w_{jk}^l}(y_i, g(x_i)) \tag{3}$$

5. Multiple layers in Neural Networks mean that $\partial C / \partial w$ needs to be evaluated by chain rule.

$\Rightarrow$ Modern Frameworks can do this process very efficiently.

1. Let $g(x_i)$ be a network output of input variables $x_i$.
2. Let $C(y_i, g(x_i))$ be the loss function for predicted output $g(x_i)$ and target output $y_i$.
3. Let $W^l = (w_{jk}^l)$ be the weights between layer $l - 1$ and $l$ where $w_{jk}^l$ is the weight between kth node in layer $l - 1$ and jth node in l.
4. Calculate the gradient of loss function in weight-space for fixed input-output pair $(x_i, y_i)$

$$\frac{\partial C}{\partial w_{jk}^l}(y_i, g(x_i)) \tag{3}$$

5. Multiple layers in Neural Networks mean that $\partial C / \partial w$ needs to be evaluated by chain rule.

$\Rightarrow$ Modern Frameworks can do this process very efficiently.

6. To optimize: Go along steepest negative gradient of $\partial C / \partial w$.

# BIOLOGICAL APPLICATIONS

| Input data | Example prediction tasks | Recommended models | Challenges |
|---|---|---|---|
| Gene sequence | DNA accessibility[14] | 1D CNNs | Repetitive regions in genome |
| | 3D genome organization[58] | RNNs | Sparse regions of interest |
| | Enhancer–promoter interactions[40] | Transformers | Very long sequences |
| Protein sequence | Protein structure[23,55] | 2D CNNs and residual networks using co-variation data | Metagenome data stored in many places and therefore hard to access |
| | Protein function[132] | Multilayer perceptrons with windowing | Data leakage (from homology) can make validation difficult |
| | Protein–protein interaction[133] | Transformers | |
| Protein 3D structure | Protein model refinement[134] | GCNs using molecular graph | Lack of data, particularly on protein complexes |
| | Protein model quality assessment[135] | 3D CNNs using coordinates | Lack of data on disordered proteins |
| | Change in stability upon mutation[136] | Traditional methods using structural features | |
| | | Clustering | |
| Gene expression | Intergenic interactions or co-expression[137] | Clustering | Unclear link between co-expression and function |
| | Organization of transcription machinery[138] | CNNs | High dimensionality |
| | | Autoencoders | High noise |

Figure 7: Different applications in Biological contexts. I [GKMJ21]

| | | | |
|---|---|---|---|
| Mass spectrometry | Detecting peaks in spectra[139] | CNNs using spectral data | Lack of standardized benchmarks[141] |
| | Metabolite annotation[140] | Traditional methods using derived features | Normalization[a] required between different datasets |
| Images | Medical image recognition[74,83] | 2D CNNs and residual networks | Systematic differences in data collection affect prediction |
| | Cryo-EM image reconstruction[60,142] | Autoencoders | |
| | RNA-sequencing profiles[143] | Traditional methods using image features | Hard to obtain large datasets of consistent data |
| Molecular structure | Antibiotic activity[73] | GCNs using molecular graph | Experimental data available for only a tiny fraction of possible small molecules |
| | Drug toxicity[54] | Traditional methods or multilayer perceptrons using molecular properties | |
| | Protein-ligand docking[39] | RNNs using text-based representations of molecular structure such as SMILES | |
| | Novel drug generation[144] | Autoencoders | |
| Protein–protein interaction network | Polypharmacology side effects[77] | GCNs | Interaction networks can be incomplete |
| | Protein function[145] | Graph embedding | Cellular location affects whether proteins interact |
| | | | High number of possible combinations |

**Figure 8:** Different applications in Biological contexts. II [GKMJ21]

▶ Presentation on 22.07.2022

- ▶ Presentation on 22.07.2022
- ▶ Prepare 30min talk

- Presentation on 22.07.2022
- Prepare 30min talk
- Everybody should be involved equally

- ▶ Presentation on 22.07.2022
- ▶ Prepare 30min talk
- ▶ Everybody should be involved equally
- ▶ How is it going so far?

QUESTIONS?

[Edu20] IBM Cloud Education. Neural networks, 2020.

[GKMJ21] Joe G. Greener, Shaun M. Kandathil, Lewis Moffat, and David T. Jones. A guide to machine learning for biologists. Nature Reviews Molecular Cell Biology, 23(1):40–55, September 2021.

[HOT06] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. Neural Computation, 18(7):1527–1554, July 2006.

[Lar18] Larhmam. Support vector machine, 2018.

[Mew20] Sandeep Mewara. codeproject.com/, 2020.

[Mit97] Tom M. Mitchell. Machine Learning. McGraw-Hill, 1997.

[MP43] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. The Bulletin of Mathematical Biophysics, 5(4):115–133, December 1943.

[Nil65] Nils J. Nilsson. Learning machines. McGraw-Hill, 1965.

[Skb09]  Skbkekas. Polynomial regression, 2009.

[Tri20]  Mayank Tripathi. Underfitting and overfitting in machine learning, 2020.