UiO : Universitetet i Oslo

Machine Learning in Computational Biology: ML Models and Algorithms

IN-BIOS5000/IN-BIOS9000

Milena Pavlović
Biomedical Informatics Research Group
Department of Informatics

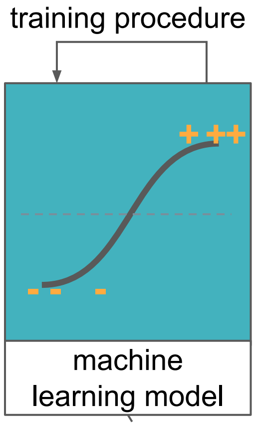
milenpa@student.matnat.uio.no

Machine learning in computational biology - outline

- Introduction to machine learning:
  - What is machine learning, types of problems, assumptions, workflow, generalization
- **Machine learning models and algorithms:**
  - **Discriminative vs generative models, supervised models (logistic and linear regression, kNN, neural networks), unsupervised models (dimensionality reduction, clustering)**
- Data representation:
  - Considerations and examples, one-hot encoding, feature engineering, representation learning
- Model comparison and uncertainty:
  - Model assessment, model selection, uncertainty, cross-validation
- Transparency and reproducibility

ML models

- We mentioned logistic regression before - a simple model for binary classification



Task: estimate function f so that f(X) = Y

Training procedure:

1. Start with some function f with some parameters
   for example, logistic regression:

$$g(\omega x + b) = \left(1 + e^{-(\omega x + b)}\right)^{-1}$$

$$f(x) = \begin{cases} 1, & g(\omega x + b) \geq 0.5 \\ 0, & g(\omega x + b) < 0.5 \end{cases}$$

Some terminology regarding ML models and algorithms

- **Learning algorithm**: a function that, given a set of examples and their labels, constructs a model, e.g., logistic regression
- **Model**: a function which was fit to the data using the learning algorithm, e.g., logistic regression with specific coefficients

Dietterich 1998

Usually model and learning algorithm are used interchangeably but they mean slightly different things

ML and statistical models overview

Classification models

Logistic Regression

Naive Bayes

Decision Tree

Support Vector Machine (SVM)

Neural Networks

K-Nearest Neighbors (kNN)

Random Forest

Boosting algorithms

Bagging algorithms

ensemble models

Regression models

Linear Regression

Polynomial Regression

Stepwise Regression

Ridge & Lasso Regression

Elastic Net

Support Vector Regression

Neural Networks

Unsupervised models

K-means clustering

Hierarchical clustering

Mixture models

Auto-encoders

Capacity of the model

- A model's capacity is its ability to fit a wide variety of functions, for instance:

  linear regression:

  $$\hat{y} = b + \omega x$$

  polynomial regression:

  $$\hat{y} = b + \omega_1 x + \omega_2 x^2$$

Capacity of the model

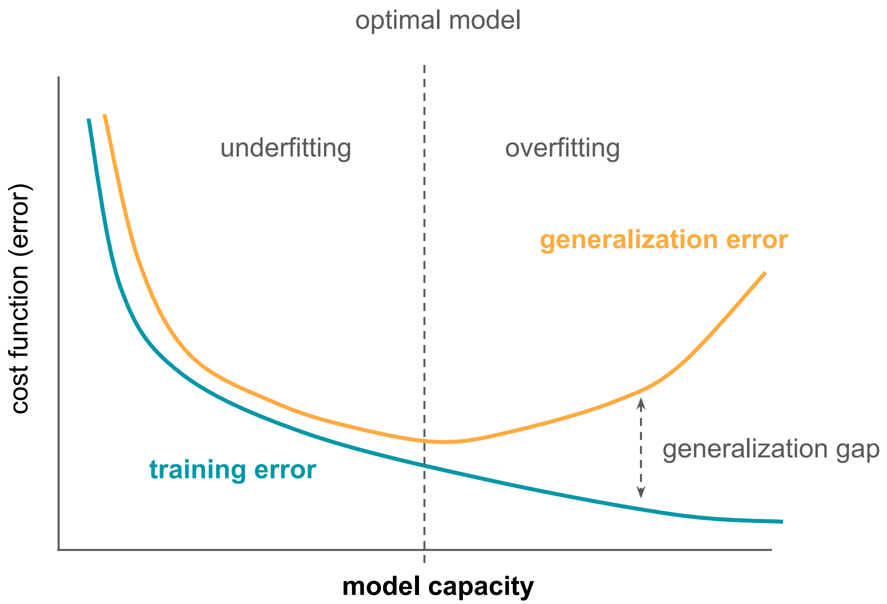
- A model's capacity is its ability to fit a wide variety of functions, for instance:

linear regression:

$$\hat{y} = b + \omega x$$

polynomial regression:

$$\hat{y} = b + \omega_1 x + \omega_2 x^2$$



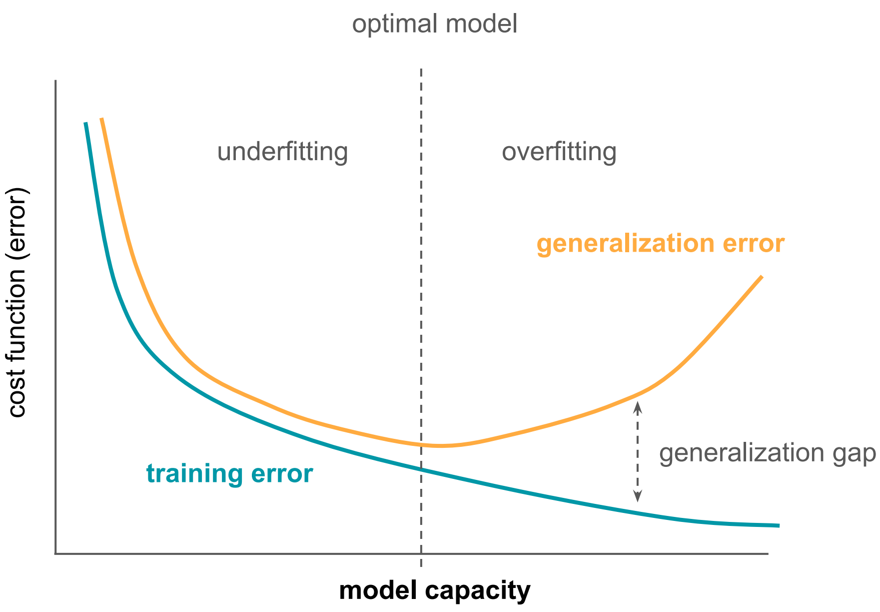
Hypothesis space of the ML algorithm

Hypothesis space is a set of functions that the algorithm can choose as the optimal solution

By choosing a hypothesis space, we control the capacity of the algorithm

Given that we chose an optimal hypothesis space, we can theoretically obtain the ideal model which knows the true probability distribution that generates the data

The minimal error achieved by the ideal model (e.g., due to noise in the data) is called Bayes error

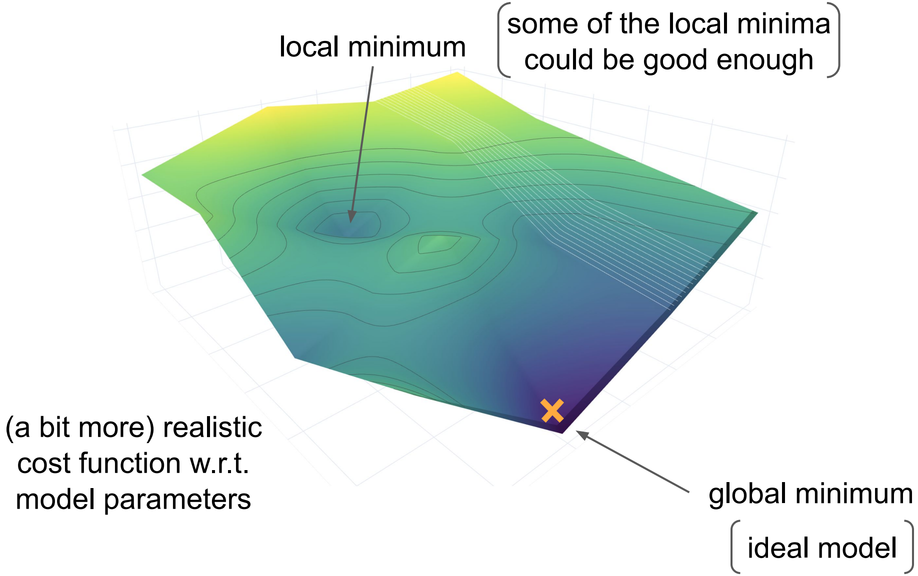


Searching through a hypothesis space using optimization algorithms

Fitting the parameters of the model is an optimization problem (there are different optimization algorithms, but one example is **gradient descent**)

We can get stuck in local minima

If the performance is good enough, we can accept that “suboptimal” model



Gradient descent minimizes the cost function to find optimal model parameters

Optimizing the cost function:

$$\text{optimal\_parameters} = \arg \min_{\text{parameters}} \{ \text{cost\_function (parameters | train data)} \}$$

We can “help” the optimization algorithm to limit the hypothesis space by imposing some restrictions on values the parameters can take

We do this by adding an additional term to the standard cost function (e.g. cross-entropy) which will increase the cost function when the model parameters do not respect our restrictions

$$\text{optimal\_parameters} = \arg \min_{\text{parameters}} \{ \text{cross\_entropy (parameters | train data)} + \alpha \ \text{regularization (parameters)} \}$$

Regularization restricts the hypothesis space

New cost function with additional term:

$$\text{optimal\_parameters} = \arg\min_{\text{parameters}} \{ \text{cost\_function (parameters} \mid \text{train data)} + \alpha \ \text{regularization (parameters)} \}$$

regularization constant
(complexity parameter)

Typical forms of regularization (also called penalty):

L1 (lasso):

$$regularization\ (parameters) = \sum_i |\, parameter_i \,|$$

L2 (ridge):

$$regularization\ (parameters) = \sum_i parameter_i^{\,2}$$

Logistic Regression

- An algorithm for binary classification:

function computing log-odds for the positive class

$$\boxed{g}(\boxed{\omega}\boxed{x} + \boxed{b}) = (1 + e^{-\boxed{(\omega x + b)}})^{-1}$$

model parameters (coefficients)

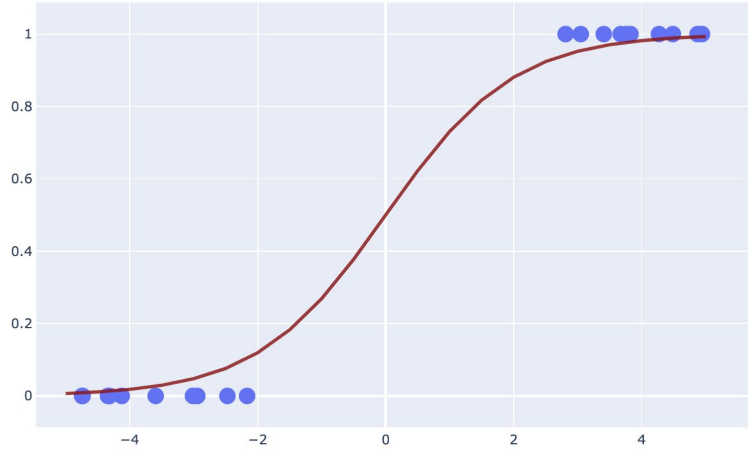
data (design matrix or feature vector)

model parameter (bias or intercept)

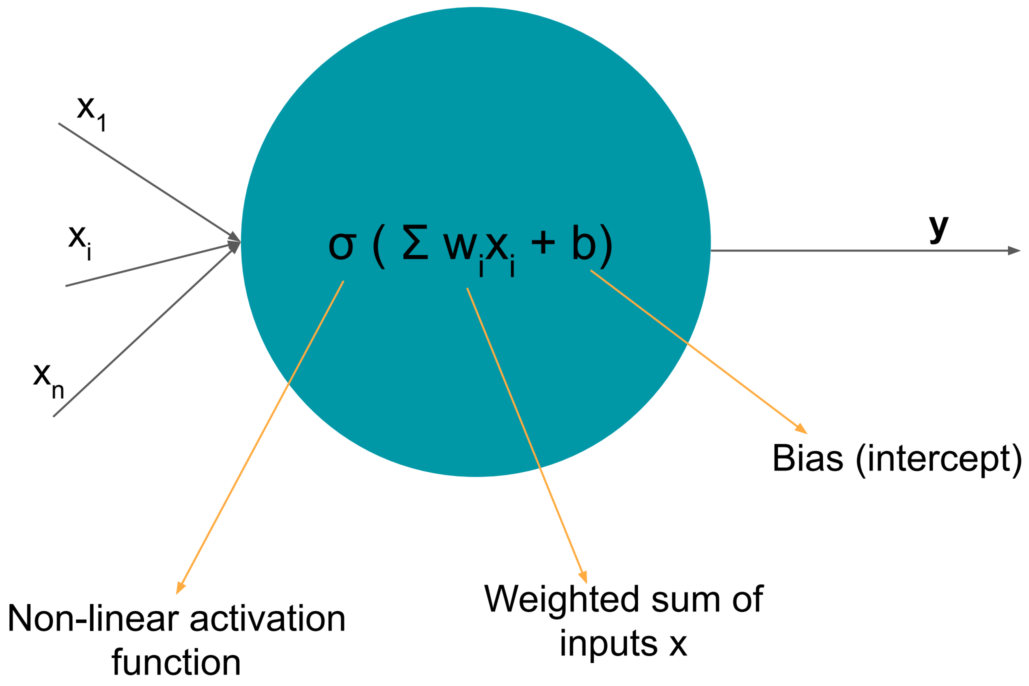
linear combination of features

function making the class prediction based on the threshold from log-odds value

$$\boxed{f(x)} = \begin{cases} 1, & g(\omega x + b) \geq 0.5 \\ 0, & g(\omega x + b) < 0.5 \end{cases}$$

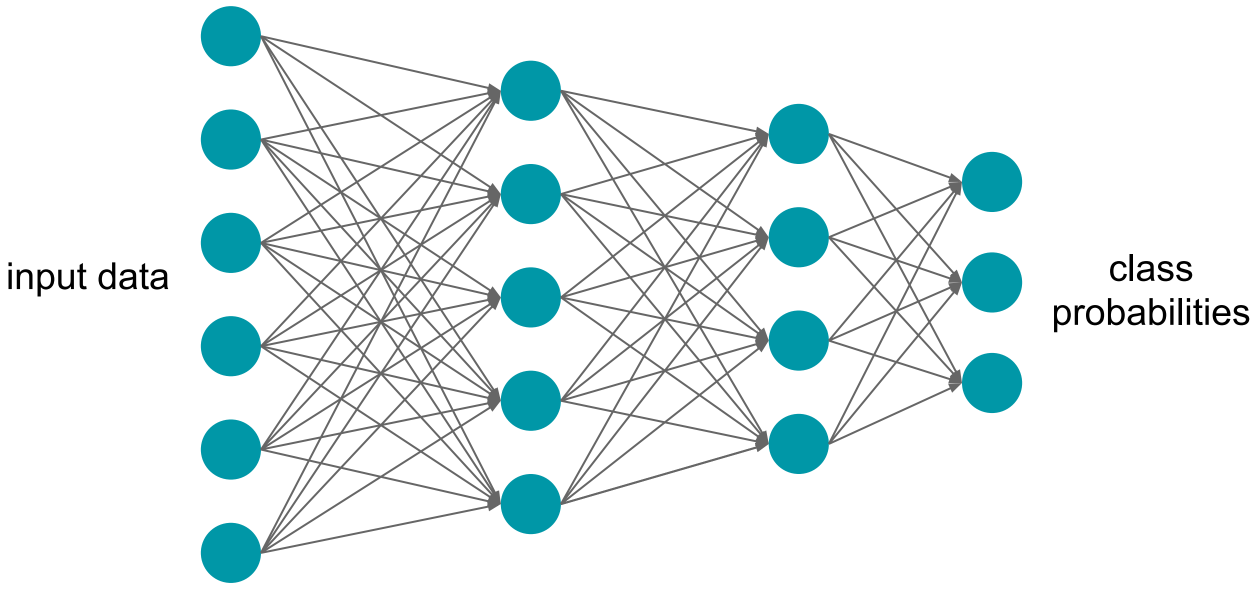


Single nodes in the neural network do something similar



Neural Networks

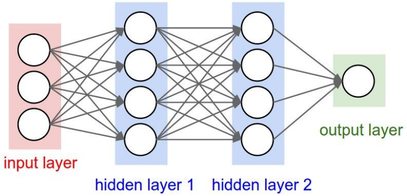
- Nodes in neural networks are organized into layers
- Number of nodes in the layer and number of layers are hyperparameters (not optimized during training, but instead set manually)
- Hierarchical structure makes them very powerful



Fully connected neural network

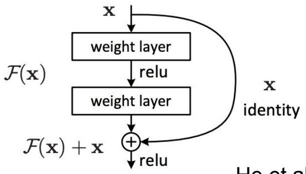
Types of neural networks

- **Fully connected networks**:
  can approximate almost any function



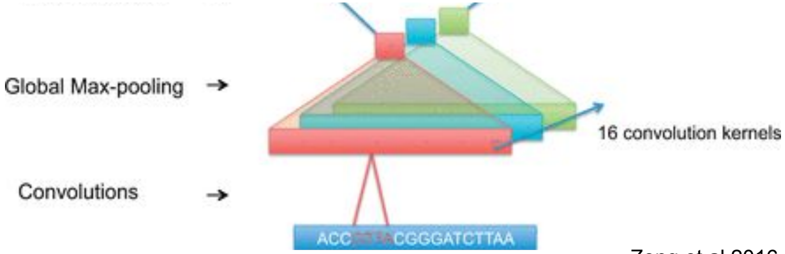
https://cs231n.github.io/neural-networks-1/

- **Residual networks**:
  can learn both simple (e.g. identity) and more complex functions



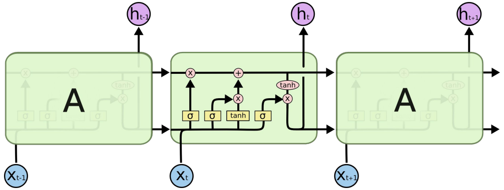
He et al 2015

- **Convolutional neural networks**:
  detect position-invariant local patterns



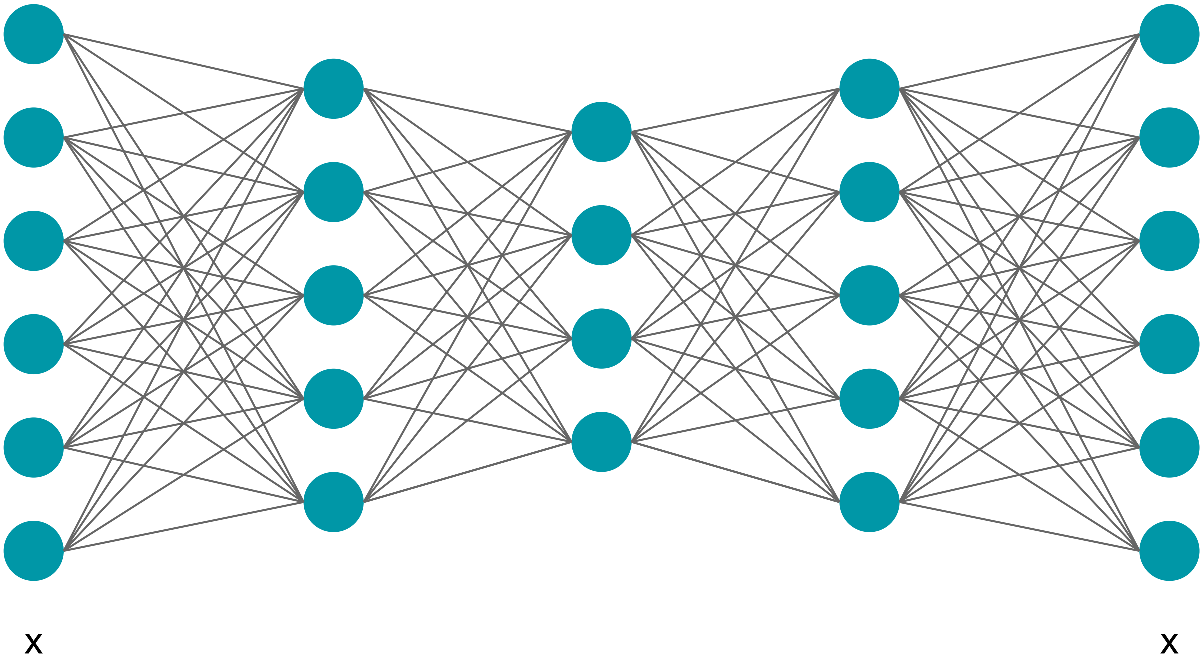
Zeng et al 2016

- **Recurrent neural networks**:
  can be Turing-complete, often used for long(er)-term dependencies in e.g., sequence data



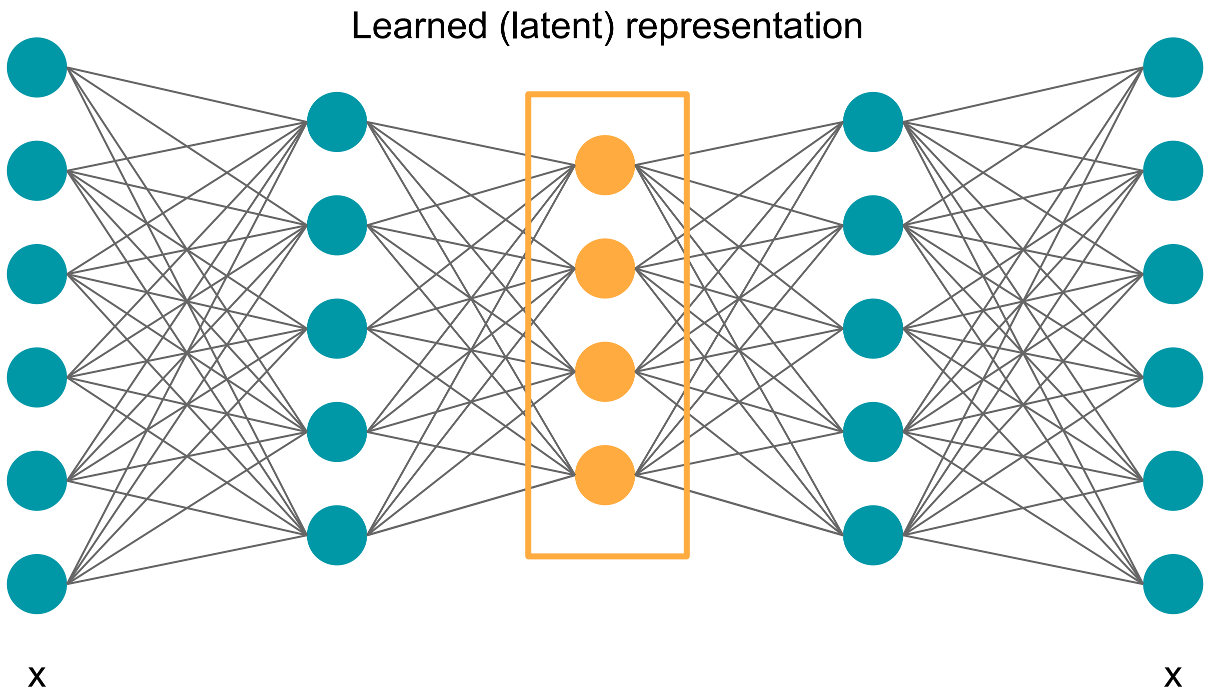
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Unsupervised algorithm - autoencoder example



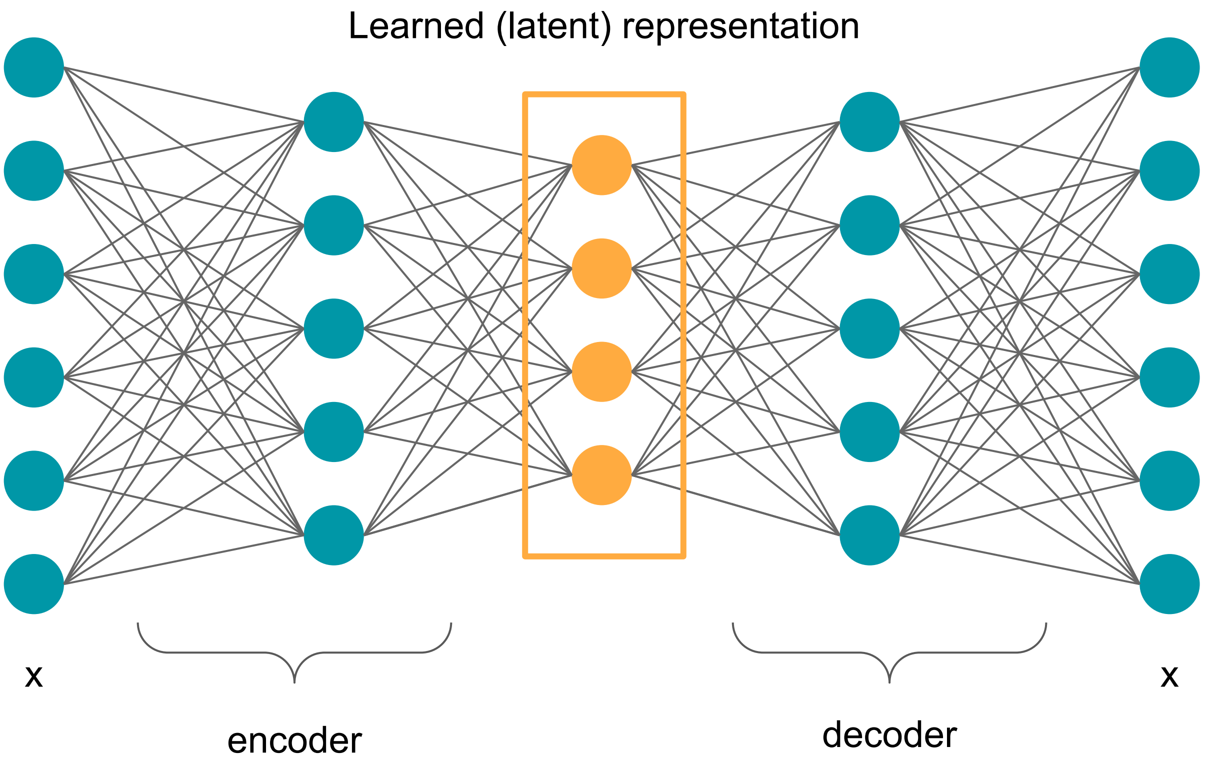
- Autoencoder is a neural network trained to attempt to copy its input to its output

Unsupervised algorithm - autoencoder example



- Autoencoder is a neural network trained to attempt to copy its input to its output while passing through a latent representation

Unsupervised algorithm - autoencoder example

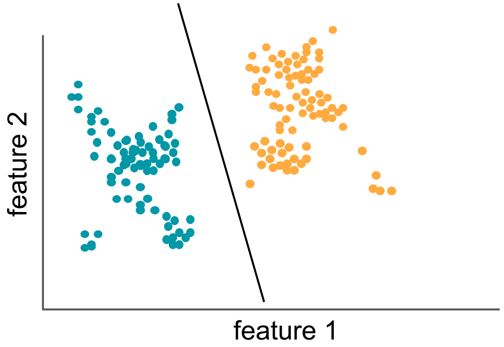


- ❑ Autoencoder is a neural network trained to attempt to copy its input to its output while passing through a latent representation
- ❑ Learned representation can have useful properties: reduced dimensionality, easy to visualize, but there are other tasks as well

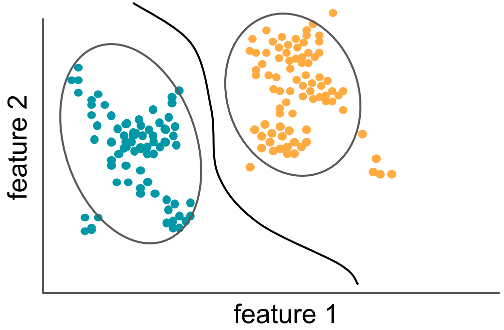
A different categorization of (classification) models

- ML models can be discriminative or generative

The model learns the conditional probability of a class given the data (but doesn't know much about the data in general)



The model learns the joint probability function of a class and the data (this is a harder problem but we learn more and we can also sample from the learnt distribution to obtain new examples)



Examples of generative models

- Generative models:
  - learn joint probability of inputs and labels in supervised setting
  - learn probability of input data in unsupervised setting → **use the model to generate new data from the same distribution**

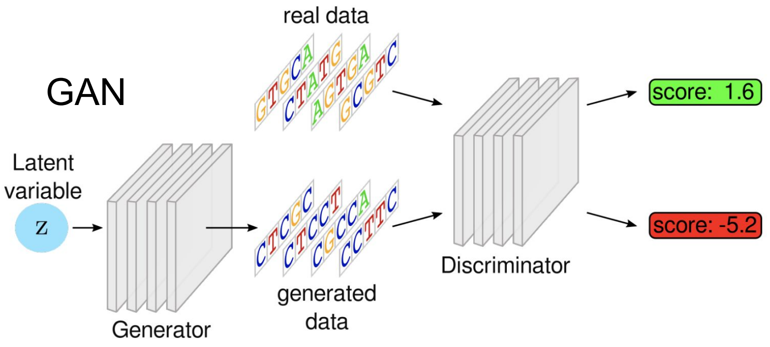
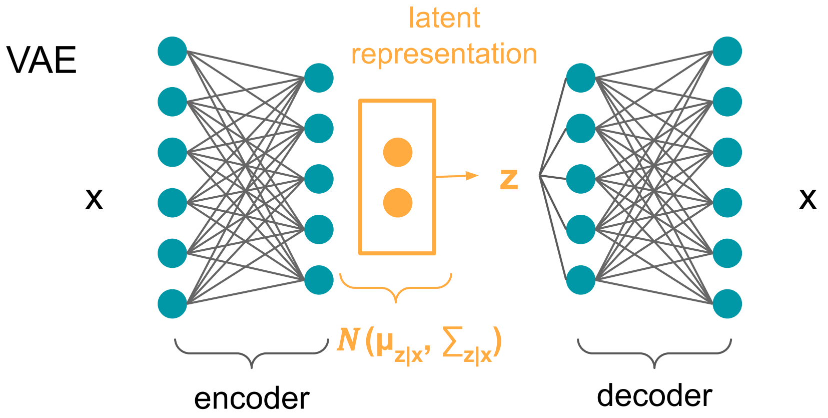


Figure: Killoran et al. 2017



References

Dietterich TG. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Comput*. 1998;10(7):1895–1923. doi:10.1162/089976698300017197

Goodfellow IJ, Bengio Y, Courville A. *Deep Learning*. MIT Press; 2016. https://mitpress.mit.edu/books/deep-learning (especially chapter 5 - Machine Learning Basics)

Killoran, Nathan, Leo J. Lee, Andrew Delong, David Duvenaud, and Brendan J. Frey. ‘Generating and Designing DNA with Deep Generative Models’. *ArXiv:1712.06148 [Cs, q-Bio, Stat]*, 17 December 2017. http://arxiv.org/abs/1712.06148.

Zeng, Haoyang, Matthew D. Edwards, Ge Liu, and David K. Gifford. ‘Convolutional Neural Network Architectures for Predicting DNA–Protein Binding’. *Bioinformatics* 32, no. 12 (15 June 2016): i121–27. https://doi.org/10.1093/bioinformatics/btw255.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. ‘Deep Residual Learning for Image Recognition’. *ArXiv:1512.03385 [Cs]*, 10 December 2015. http://arxiv.org/abs/1512.03385.