Notes on

Machine Learning Lecture

Taught by
Prof. Dr. Joni Dambre
Prof. Dr. Tom Dhaene

Created by
M.Sc. Student
Karina Chiñas Fuentes

Universiteit Gent

August 14, 2023

Weekly Content

DISCLAIMER

These are my notes and associated further reading that regards my understanding of Machine Learning based on the Universiteit Gent 2022 winter lecture. Non-cited statements come from the lecture. I also make notes to myself. If the reader finds something useful, I encourage them to investigate further. Non-cited charts were created by me, and their nonprofit-related usage is allowed with a citation.

The following text is from me to me; to prepare myself for the examination and enhance my knowledge of related topics. The empty spaces are for me to write once I print these notes.

Before you write:

1. What are the keywords?
2. What are the assumptions of the model or algorithm?
3. What are the main take away?
4. Flow Charts?
5. Advantages and Disadvantages
6. Are they linear or non-linear transformations / models

Recall that the Recap slides from Prof. Dr. Joni Dambre also helps.

1 First Steps in Supervised Learning

1.1 What is Machine Learning?

Machine Learning is the field that generates algorithms that learn from data so they can predict or make decisions based on their learning. One aims to **build a model for data by optimizing the performance criterion using training data**. See Figure 1 for the different fields surrounding machine learning.

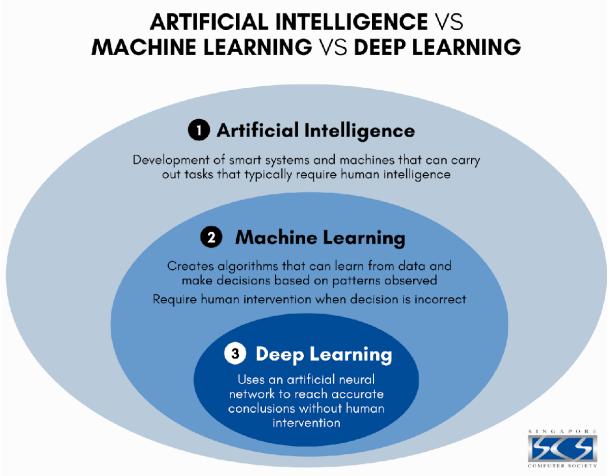


Figure 1: Surrounding fields in Machine Learning. Image from [1].

To understand machine learning algorithms, it is necessary to know about Probability Theory and Statistics. It is also desired to have efficient programming knowledge, so this is an interdisciplinary field.

Learning algorithms are implemented when no rules are available to establish the relationship between the input and the output. It is also possible that the algorithm is required to adapt to different environments.

1.2 Types of Machine Learning

There are three main different types of machine learning; these are depicted in Figure 2.

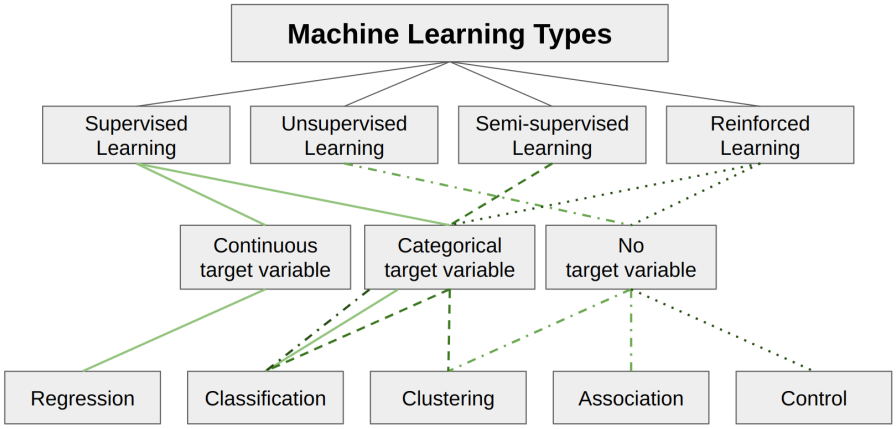


Figure 2: Different types of machine learning. Image inspired from [2].

- **Supervised**: here, the expert gives the algorithm the output data (as labels for classification) so that the input-output relationship can be modeled. The type of tasks covered here are:
  - Regression: when the output is numerical.
  - Classification: when the output is categorical.
- **Unsupervised**: the aim is to learn without an output; this approach allows us to model the structure of the data or the underlying profile of the data; they can become *discriminative* or *generative*[1]; for instance, Gaussian mixture models[2], random forests, and neural networks are discriminative models. Generative models are often used as pre-processing in deep learning [4].

[1] The algorithm focuses on devising the decision boundary (without underlying assumptions on data) or on modeling the joint distribution of inputs and outputs, respectively.
[2] See more in [3].

– Clustering: hierarchic and nonhierarchic are their sub-branches; more might be investigated. Clustering can be both discriminative and generative.

– Association / Transformation of Features: the aim is to uncover latent variables or compress information. This can either improve or worsen the interpretability of the model, e.g. Principal Component Analysis (PCA); see Figure 3.

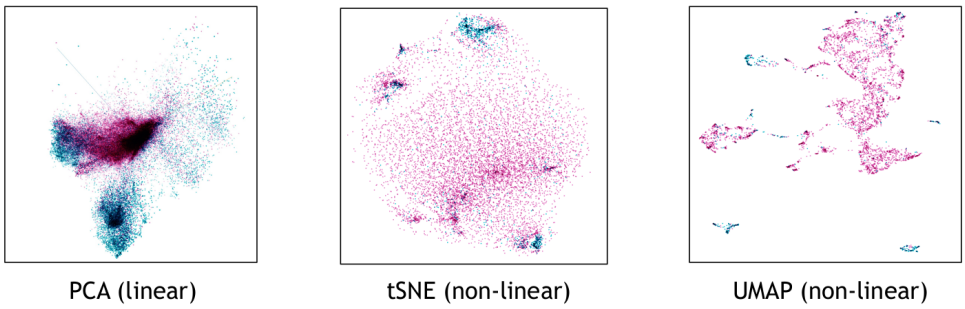


Figure 3: Different types of feature transformation, showing PCA (left), t-Distributed Stochastic Neighbor Embedding (center), and Uniform Manifold Approximation and Projection (right). Image from lecture 1.

- **Semi-supervised Learning**: as the name suggests, this is a mixture of the previous two and is typically implemented when the cost of labeling is high [2].
- **Reinforced Learning**: this method uses gathered information from the interaction with the environment to maximize reward and reduce risks [2]; this can be implemented for real-time decisions.

1.3 Supervised Learning

The notation used in the lecture is the following[3]:

- Ground truth: $\mathbf{y} = f(\mathbf{x}) + \varepsilon$
- Training data: $\mathbf{X}, \mathbf{Y} : (\mathbf{x_1}, y_1), \dots (\mathbf{x_N}, y_N)$
- Hypothesis: $f(\mathbf{x}) \sim g(\mathbf{x}, \mathbf{X}, \boldsymbol{\theta}) + \varepsilon$; best model that can be learned from a certain model family.
- Loss function: $\mathcal{L}(\boldsymbol{\theta} \mid \mathbf{X})$
- Learning: $\hat{\boldsymbol{\theta}} = argmin_\theta \left(\mathbb{E}\left[\mathcal{L}(\boldsymbol{\theta} \mid \mathbf{X})\right]\right)$; minimize expected loss on new data.

Notice that $f(\mathbf{x})$ is unknown. In general, finding the optimal parameters that will minimize the expected loss function on new data is necessary. Optimal parameters depend on the assumption made on data. Finding non-parametric algorithms in supervised learning for regression and classification is also possible, e.g., K-nearest neighbors (KNN). See Figure 4 for a better comparison.

! Algs: KNN (regression and classification), decision trees (regression and classification), linear discriminate (regression and classification), and non-linear feature expansion (regression).

Consider mentioning the training/inference time and memory scale w.r.t. dataset size when describing an algorithm. These issues become relevant once the quality of the data has been addressed.

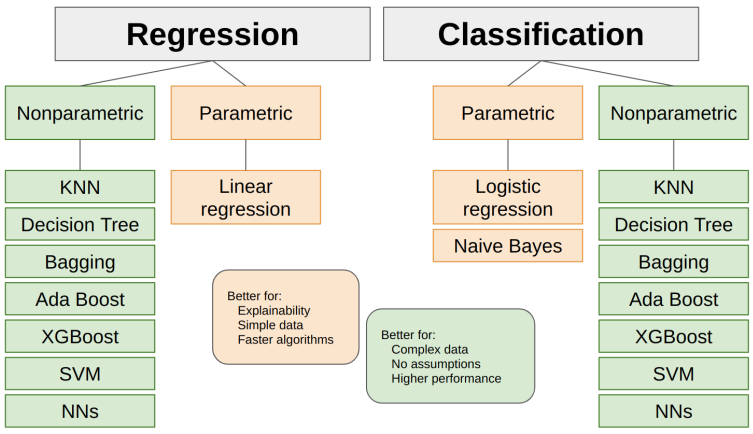


Figure 4: Types of algorithms in supervised learning and some examples. Image inspired from [6].

[3]However, the nomenclature is not always maintained; therefore, you might find the notation following in [5]. To avoid confusion, it is explained in the text what variables mean.

1.4 Evaluating Models and Conditions for Generalization

Machine learning aims to achieve **generalization**; the algorithm should perform well when new data is introduced. To understand how this is feasible, important concepts will be introduced.

- **Loss function**: this is among the most important metric in machine learning. This function allows the user to tell how good a model's prediction is, based on the *the loss*. This transforms the learning into an optimization problem by defining a loss function and optimizing it to its minimum. More on this shall be explored [7]. ! Different loss functions, different predictions.
- **Empirical Risk Minimization** (ERM): the understanding of ERM permits us to understand the limits of an algorithm, which also allows the development of practical skills in machine learning [8]. The concept of empirical risk arises from the fact that the user cannot access the *true error* since the algorithm only receives a sample of an unknown distribution from the data. However, estimating the *training error* – the error the algorithm incurs over the training sample [9] is possible. In short, ERM is the search for a predictor (or model) that minimizes the training error; this can be depicted in the following expression

$$\hat{\boldsymbol{\theta}} = argmin_{\theta} \left(\frac{1}{N} \sum_{j=1}^{N} \mathcal{L}(x_j \mid \boldsymbol{\theta}, \mathbf{X}) \right). \tag{1}$$

  Remember that the model represents your **hypothesis**. ERM: the search for the best model that minimizes training error.
- **Test Error**: because the ERM does not tell the user how well the algorithm is capable to predict on new data, what is typically done is to split the available data such that, once the algorithm has been trained (found $\hat{\boldsymbol{\theta}}$), the algorithm can be assessed on the performance on the *test data*. ! WARNING: depending on how you split the data, how likely you are introducing *lekeage* in your model.
- redo **IID Assumption**: Identically and Independent Assumption:
  - Indetically: training, validation, test, and new incoming data are drawn from the same underlying joint distribution. In other words, all datasets must have the same statistical properties: mean, standard deviation, and other characteristics [10].

    Violating this assumption creates

    * prediction bias; predictions are not characteristic of the real world due to lack of representative groups, e.g., training facial recognition on engineering students: population not very diverse, very few women, no children, no elderly people. This can also result from lack of balanced classes.

      Feature and label bias can be considered a violation of the "identically distributed" part of the i.i.d. assumption.
  - Independent: no subgroups of samples correlated in any type of data subset (training, validation, and test); in other words, any data point should not provide information on the occurrence or value of another data point [10]. Usually, The correlation is attributed to **latent varibales**, a.k.a. unknown or unobserved variables.

    Violating this assumption creates

    * bias;

  Violation of any assumption leads to poorer generalization, and they can be interconnected. !!! I think what is important in this discussion is identifying the source of a violation rather than the independent consequence because the violation of each might result in a similar consequence. FOCUS ON THE SOURCE RATHER THAN THE CONSEQUENCE.

  —

  For instance, the characteristic clusters **in STDB5** for the spherical and non-spherical tokamaks violate the identical assumption. This might not necessarily mean that spherical tokamaks must have their own scaling law (?). In reality, there is no continuum in aspect ratio to effectively assess whether they are *identical* machines. This implies too much money to discover.

1.5 Model Selection: hyperparameter analysis

When one changes the hyperparameters of an algorithm, it can be considered a different model than before; for instance, the number of features one uses is a hyperparameter. How can one tell which model is better compared to another? It is recommended to follow the diagram shown in Figure 5, left.

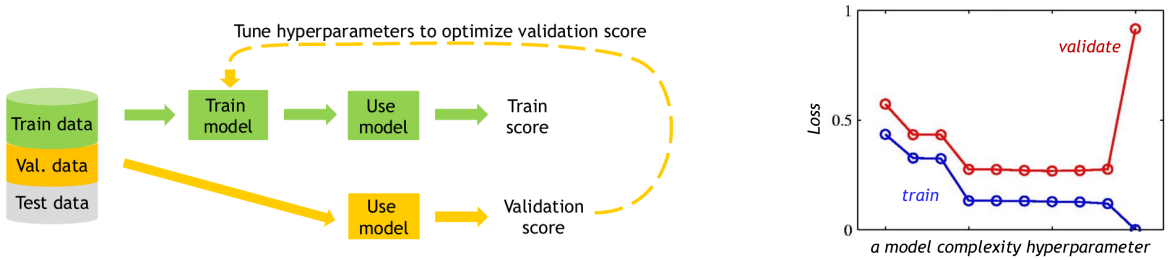


Figure 5: Left: data split for model assessment, all data subsets are assumed to follow IID assumption. Right: validation curve; if the IID assumption is not fulfilled, the curves will not look similar; this is why it is recommended to look at this plot at the beginning. Sometimes, the validation curve is plotted with accuracy or another metric, like F_1-score. Images from lecture 1.

When you tune the model's hyperparameters, you also change the model complexity; the validation curve is assessed from this analysis. The validation curve plots a model performance metric vs. the algorithm's complexity [11]. The typical behavior is that the validation error goes down as the complexity increases, but the error goes up again at some point; see Figure 5, right. **More on this will be explained when the bias-variance decomposition in a model is studied.** Once the ideal model has been found, the train and validation datasets are merged to obtain the overall train and test errors without further modification. This is depicted in Figure 6

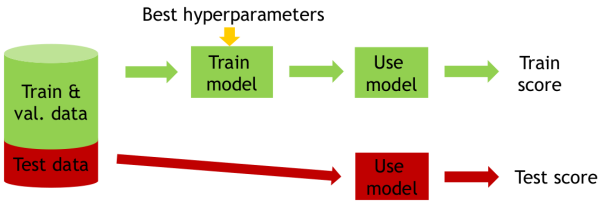


Figure 6: Data split for model assessment, final stage. Image from lecture 1.

If the validation and training data are too small, the model will likely have a poor choice of hyperparameters and might overfit. Again, beware of how you split the data, as you might introduce a source of leakage.

Another discussion brought on the analysis of model complexity is the choosing of features. In general, it is not ideal to have too many features due to what is known as *the curse of dimensionality*. The reasoning is that a higher number of features implies a more complex model, and the more complex the model, the more data it requires to perform well. For instance, consider the case of fitting data with a polynomial of degree d with F total number of features. The complexity of the said model is $\mathcal{O}(F^d)$; from this, it is possible to observe that the selection of features becomes a serious matter for optimal performance, directly affecting its generalization capability.

Here are some forms of **regularization** for optimal model selection:

- Spend time searching for the optimal subset of features to avoid the "curse of dimensionality".
- If possible, reduce the number of hyperparameters (this is a matter in neural networks).
- Constrain parameter space: the idea is that you assess for a region in parameter space where you know solutions will lead to an optimal generalization, e.g. L_1 and L_2 regularization, more on this in the incoming chapter.
- When working with optimization algorithms, such as gradient descent, avoid over-tuning in training data (e.g. early stopping).

1.6 Summary

Key words: empirical risk minimization (ERM), validation curves, curse of dimensionality.

2 Supervised Parametric Linear Models and Introduction to SVM

Three central concepts must be understood when discussing parametric models: loss function, regularization, and optimization [5]. To understand these concepts, let us have a look at the equation that *models* the true input-output relationship

$$y = f(\mathbf{x}) + \varepsilon. \tag{2}$$

Depending on the assumptions, is the algorithm; for instance, if $f(\mathbf{x}) = \boldsymbol{\theta}^T\mathbf{x}$ is linear, and the noise is Gaussian $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, you retrieve ordinary least squares [5]. Derive the previous statement:

Of course, $f(\mathbf{x})$ can be non-linear, and other assumptions can be imposed over the irreducible noise, depending on the problem; for instance, the assumption of $\varepsilon \sim$ Laplace distribution, one obtains the *absolute error loss* [5]. However, if one keeps the assumption of Gaussian noise, one will obtain a Gaussian joint likelihood [5], which eases the interpretation and working of the model's predictions.

The main objective of the learning algorithm is to learn enough from the training data to properly predict or make decisions on unseen data; this is known as *generalization*. A model may also learn the noise from the training data rather than the underlying properties of it; this is known as *overfitting*. The loss function is implemented to prevent overfitting so that the model does not mimic the training data.

2.1 Loss Functions

Recall that, the learning of an algorithm means that one is dealing with an optimization algorithm [5] of the parameters $\boldsymbol{\theta}$. As previously mentioned, the optimization problem is formulated as follows

$$\hat{\boldsymbol{\theta}} = argmin_\theta \left(\frac{1}{N} \sum_{t=1}^{N} \mathcal{L}\left(y_i, f_\theta(\mathbf{x_i})\right) \right), \tag{3}$$

here, one observes

- the **loss function** $\mathcal{L}(y_i, f_\theta(\mathbf{x_i}))$, and
- the **cost function** $J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{t=1}^{N} \mathcal{L}\left(y_i, f_\theta(\mathbf{x_i})\right)$.

It is possible that a solution to the optimization problem is not exactly and is not directly computable; this is particularly common in non-linear functions, e.g. no closed-form solution exists for logistic regression. Therefore, a solution might be found through numerical optimization [5]. However, the ultimate goal is not solving Eq. (3), but rather

$$\hat{\boldsymbol{\theta}} = argmin_\theta \left(E_{new}(\boldsymbol{\theta}) \right), \tag{4}$$

with $E_{new}(\boldsymbol{\theta}) = \mathbb{E}_* \left[E(\hat{y}, y_*) \right]$, being the expected error on predictions w.r.t. unseen data. Nevertheless, one does not have access to this information; for this reason that **the loss minimization is the representative for generalization** [5]. Therefore, it is of crucial importance that the machine learning practitioner understands that the training objective, Eq. (3), is just a proxy for the actual objective, Eq. (4). The authors in [5] stress on the following observations

- optimization accuracy $\neq$ statistical accuracy,
- loss function $\neq$ error function, and
- explicit vs implicit regularization (explain *asymptotic minimizer* and *strictly proper*[4] functions).

[4] Hint: the downside of Hinge loss; remedy: squared Hinge loss. Do not forget to see p.105 in [5].

It is important to understand that selecting the loss and error functions comes as part of the model's design. There is no right nor wrong selection, but rather, what represents best your data. For instance, one has

- linear regression: linear in the parameter model and $\mathcal{L}$ being the squared error loss; and,
- support vector classification: linear in the parameter model and $\mathcal{L}$ being the Hinge loss [5].

IMPORTANT: a loss function is said to be robust if the training data containing a considerable amount of outliers only has a minor impact on the learned model [5].

2.1.1 Regression

As already discussed, some loss functions come naturally from the assumptions made on the data, such as the *squared error loss*

$$\mathcal{L}(y, \hat{y}) = (\hat{y} - y)^2, \tag{5}$$

coming from a Gaussian noise in Eq. (2); and, the *absolute error loss*

$$\mathcal{L}(y, \hat{y}) = |\hat{y} - y|, \tag{6}$$

if the noise in Eq. (2) is $\varepsilon \sim L(0, b_\varepsilon)$ Laplacian [5]. However, not all loss functions come from an instinctive derivation from data assumption; some are designed to achieve a superior generalization. For instance, the *Huber loss*

$$\mathcal{L}(y, \hat{y}) = \begin{cases} \frac{1}{2}(\hat{y} - y)^2 & \text{if } |\hat{y} - y| < 1 \\ |\hat{y} - y| - \frac{1}{2} & \text{else.} \end{cases} \tag{7}$$

This is a combination of the previous two; this is because the absolute error is moss robust to outliers than the squared error [5]. Another variation to this idea is depicted with the *ϵ-insensitive loss*

$$\mathcal{L}(y, \hat{y}) = \begin{cases} 0 & \text{if } |\hat{y} - y| < \epsilon \\ |\hat{y} - y| - \epsilon & \text{else,} \end{cases} \tag{8}$$

where ϵ is left to be chosen for the design, notice that the *ϵ-insensitive loss* leaves a tolerance width 2ϵ around the observed y and, outside the region, behaves like the absolute error loss [5]. This loss function proves particularly useful for support vector regression [5]. See Figure 7 to compare the robustness of the functions mentioned in this section.

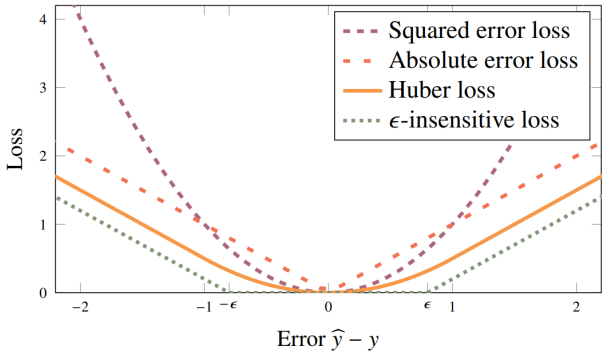


Figure 7: Different loss functions for regression given a specific error function. Image from [5].

2.1.2 Classification

An instinctive loss function for binary classification could have similar behaviour to the Heaviside function; there is one known as the *misclassification loss* written as

$$\mathcal{L}(y, \hat{y}) = \begin{cases} 0 & \text{if } \hat{y} = y \\ 1 & \text{else.} \end{cases} \tag{9}$$

Despite that one might have the goal of suppressing the misclassification rate the most, this is not the commonly chosen loss function for three main reasons (explain). The usage of the *cross-entropy loss*

$$\mathcal{L}(y, g(\mathbf{x})) = \begin{cases} \ln(g(\mathbf{x})) & \text{if } y = 1 \\ \ln(1 - g(\mathbf{x})) & \text{if } y = -1 \end{cases} \tag{10}$$

is more used. Here, $g(\mathbf{x})$ is the probability that the observation $\mathbf{x}$ belongs to a class; this allows a complete statistical description of the output-input conditional distribution [5]. To understand other families of loss functions, it is essential to comprehend the concept of **margins**. In general, it is defined that **the margin of a classifier for a data point** $(\mathbf{x}, y)$ **is** $y \cdot f(\mathbf{x})$; so that if the margin has the same sign, the classification is considered correct, otherwise incorrect [5]. This construction helps when the classifier does not have a probabilistic interpretation. Still, rather it is just being represented with an underlying function $f(\mathbf{x})$; it can be seen as a measure of certainty in a prediction [5] (!!! think: similarity between error in regression loss functions and margin in classification loss functions). From the construction of the margin, it is possible to assign a small loss to a positive (correct classification) margin and a considerable loss to a negative margin. Figure 8 shows diverse loss functions for classification with different robustness to outliers.

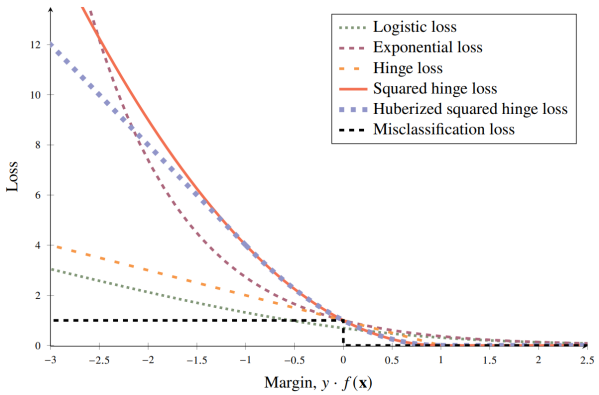


Figure 8: Different loss functions for classification given the margin. Image from [5].

Write the expressions for the different loss functions shown in the image with their respective properties and algorithms usage. Explain losses for multiclass classification.

2.2 Learning parameters for Specific Cases

Explain the solution and main characteristics when minimizing the loss functions for regression and classification using MSE (or SSE) and log-loss, respectively. Explain gradient descent and compare with coordinate descent.

2.2.1 DEMO: Gradient descent for linear regression notebook

Make your notes.

2.3 Overfitting and Explicit Regularization

As mentioned, generalization is when an algorithm performs well on unseen data. Overfitting happens when proper generalization is not achieved, and the algorithm only performs well on the training dataset; there are many causes of overfitting, one being a model being too complex (e.g. number of features being used).

The goal of regularization in parametric models is that: "if a model with small values of the parameter $\hat{\boldsymbol{\theta}}$ fits the data almost as well as a model with larger parameter values, the one with small parameter values should be preferred." [5]

Regularization is the act of preventing overfitting. For implicit regularization, one can modify the expression of the loss function; for instance, **Ridge regression** or $\mathbf{L}^2$ **regularization** is the addition of a function that penalizes high weights in polynomial regression (because high values of weights mean high complexity algorithm). The resultant expression is the following

$$\mathcal{L}(y, \hat{y}) = \frac{1}{2}(\hat{y} - y)^2 + \lambda \cdot ||\mathbf{w}||_2^2. \tag{11}$$

With $\lambda \geq 0$, one always has a unique solution to regression – the best value is obtained through cross-validation. $\lambda = 0$ gives OLS solution. Notice that the modified equation presents a trade-off between having the perfect fit and enforcing the regressor parameters to being close to zero; the greater λ, the closer to zero the $\hat{\boldsymbol{\theta}}$ values. Given the loss function with Ridge regression, one obtains [5]

$$\hat{\boldsymbol{\theta}} = \left(\mathbf{X}^T\mathbf{X} + n\lambda\mathbf{I}_{p+1}\right)^{-1}\mathbf{X}^T\mathbf{y}, \tag{12}$$

with n denoting the number of observations in the dataset. Another type of regularization is known as **LASSO** or $\mathbf{L}^1$ **regularization**, and it is of the following form

$$\mathcal{L}(y, \hat{y}) = \frac{1}{2}(\hat{y} - y)^2 + \lambda \cdot ||\mathbf{w}||_1, \tag{13}$$

where $||\mathbf{w}||_1$ is the 1-norm $||\mathbf{w}||_1 = |w_1| + |w_2| \ldots + |w_p|$ [5]. It is worth noting that there is no closed-form solution to this loss function; numerical optimization algorithms must be used to solve the LASSO regression. The effect of this type of regularization is that it can *switch-off* some coefficients (by setting specific weights to zero) and provide sparse solutions; it is for this reason that, sometimes, this is a method of feature selection [5].

Following images from the book [5].

General Explicit Regularisation

L^1 and L^2 regularisation are two common examples of what we refer to as explicit regularisation since they are both formulated as modifications of the cost function. They suggest a general pattern on which explicit regularisation can be formulated:

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \underbrace{J(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y})}_{\text{(i)}} + \underbrace{\lambda}_{\text{(iii)}} \underbrace{R(\boldsymbol{\theta})}_{\text{(ii)}}. \tag{5.26}$$

This expression contains three important elements:

(i) the cost function, which encourages a good fit to the training data;

(ii) the regularisation term, which encourages small parameter values; and

(iii) the regularisation parameter λ, which determines the trade-off between (i) and (ii).

Method	***What is optimisation used for?***			***What type of optimisation?***			
	Training	*Hyper-parameters*	*Nothing*	*Closed-form**	*Grid search*	*Gradient-based*	*Stochastic gradient descent*
k-NN							
Trees							
Linear regression							
Linear regression with L^2-regularisation							
Linear regression with L^1-regularisation							
Logistic regression							
Deep learning							
Random forests							
AdaBoost							
Gradient boosting							
Gaussian processes							

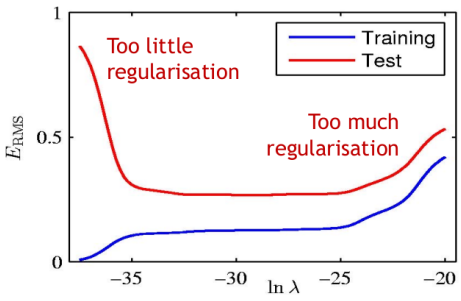
**including coordinate descent*

Figure 9: Images from [5]

Generalisation: perform well on unseen data!

Train model on training data

More complex model: *training error* decreases



Evaluate model on separate test data: at some point test error goes up again.

⇒ *Overfitting to the training data!*

⇒ *Select best degree based on unseen data!*

Figure 10: Image from lecture 2

2.3.1 DEMO: Linear regression regularization notebook

Make your notes.

2.4 Classification: Linear Support Vector Machine

2.5 Classification: Soft Margin Support Vector Machine

2.6 Summary

Keywords: asymptotic minimizer, strictly proper, margins, optimization, open- and closed-form solutions.

3 Reasoning About Models and Data

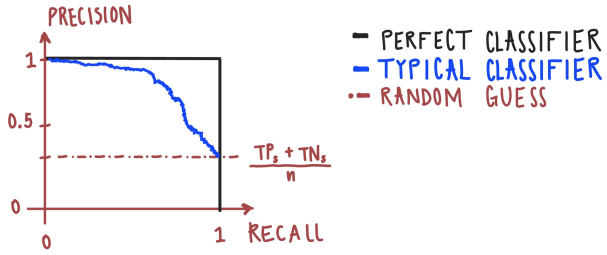
3.1 Error Functions for Model Selection

Describe the confusion matrix and why it is needed.

For the following, add comments based on slides.

Table 1: Some of the obtainable metrics from a confusion matrix [5], [12]. Remember that n is a dataset's total number of observations.

Metric	Formula	Description
Misclassification Rate	$\frac{\text{FNs} + \text{FPs}}{n}$	Fraction of predictions being incorrect
Accuracy	$\frac{\text{TNs} + \text{TPs}}{n}$	Complement of misclassification rate
Precision	$\frac{\text{TPs}}{\text{TPs}+\text{FPs}}$	Fraction of predicted positives actually being positive
Recall	$\frac{\text{TPs}}{\text{TPs}+\text{FNs}}$	Fraction of actual positives correctly predicted
Fall-out	$\frac{\text{FPs}}{\text{FPs}+\text{TNs}}$	Probability of false alarm
Specificity	$\frac{\text{TPs}}{\text{FPs}+\text{TNs}}$	Compliment of fall-out
False discovery rate	$\frac{\text{FPs}}{\text{FPs}+\text{TPs}}$	Fraction of incorrect positive predictions
False negative rate	$\frac{\text{FNs}}{\text{FNs}+\text{TPs}}$	Fraction of actual positive incorrectly classified
False omission rate	$\frac{\text{FNs}}{\text{FNs}+\text{FPs}}$	Fraction of incorrect negative relative to tall incorrect classifications
Prevalence	$\frac{\text{FNs} + \text{TPs}}{n}$	Proportion of actual positive instances in the dataset
F_1-score	$\frac{2 \cdot precision \cdot recall}{precision + recall}$	Harmonic mean of precision and recall
F_β-score	$\frac{(1+\beta^2) \cdot precision \cdot recall}{\beta^2 (precision + recall)}$	Used to account that recall is β-times as important as precision



Explain RoC curve.

Explain hypothesis testing and its importance.

3.2 Bias and Variance

For this discussion, the **IID assumption is conserved**. This discussion centers on *What happens if you train a model multiple times on different IID data sets of the same size?* The answer is that one gets **different models**, with the difference being more notorious if the complexity of the model is high; however, the difference decreases if the amount of data increases in each case. This is depicted in Figure 11

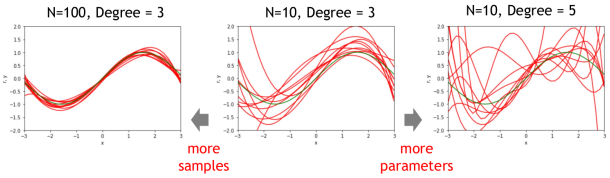


Figure 11: Image from lecture 3.

Explain what bias and variance are **in statistics**.

An **estimator** or a model $g()$ is an instance of a model family being fed with a fixed number of observations in the dataset.

Derive an estimator's bias and variance decomposition (see the summary slide of bias and variance).

Explain the theoretical plot of generalization error [5]:

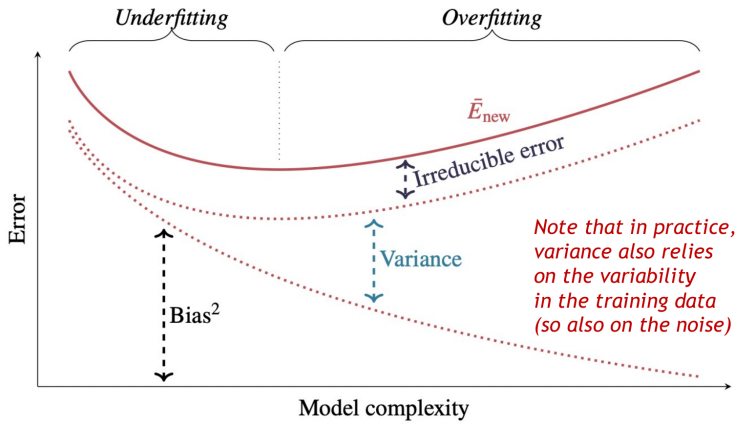


Figure 12: Image from [5].

The trade-off between model complexity and dataset size opens the discussion on

- **MODEL BIAS**: this theoretical concept cannot be measured because the user cannot access the underlying data distribution. This characterizes being
  - **the expectation** across various models trained with different data subsets of the expected model error on unseen data, compared to the ground truth.
  - **the capability** of the chosen estimator to approximate the ground truth.

  High bias implies that the model is too simple to extract suitable information from the features; this is also referred to as **underfitting**. The result of high bias can be a high observation of error; however, this could also be from a large amount of irreducible error (see plot above), meaning that features do not carry the necessary information for estimation.
- **MODEL VARIANCE**: this theoretical concept cannot be measured because the user cannot access the underlying data distribution. This characterizes being
  - **the variance** across various models trained with different data subsets of the expected model error on unseen data, compared to the ground truth.
  - **the sensitivity** of the chosen estimator to variability in training data.

  When high variance is observed, this is because the model is **overfitting** the training data, so it is suggested to either obtain more data or make the model less complex.

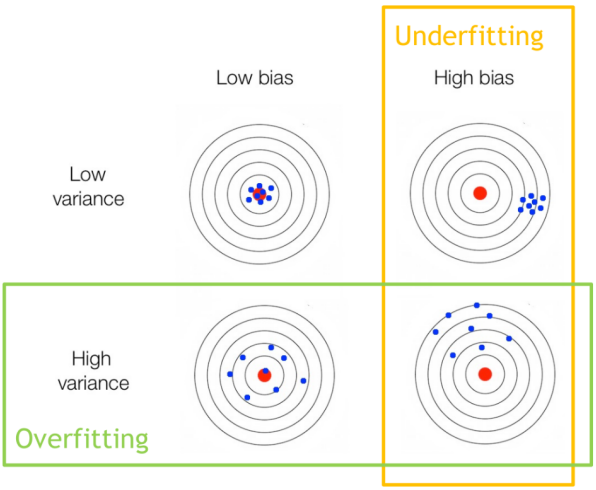


Figure 13: In the bull's eye plots, each blue dot represents the average error on unseen data of a model trained on a different dataset; the red mark represents the perfect model. Without better data, model generalization error trades-off between high variance and low bias and vice versa. Image from lecture 3.

Bias and variance trade-off if you keep the amount of data and the model type fixed but tune the model complexity.

A model can be complex enough to overfit the training data but still be unable to capture the ground truth due to a large irreducible error; hence, **it is possible to have both in a model**.

Figure 14 considers how the generalization error and variance change as the dataset size increases for a theoretical example with known ground truth and noise-free error.

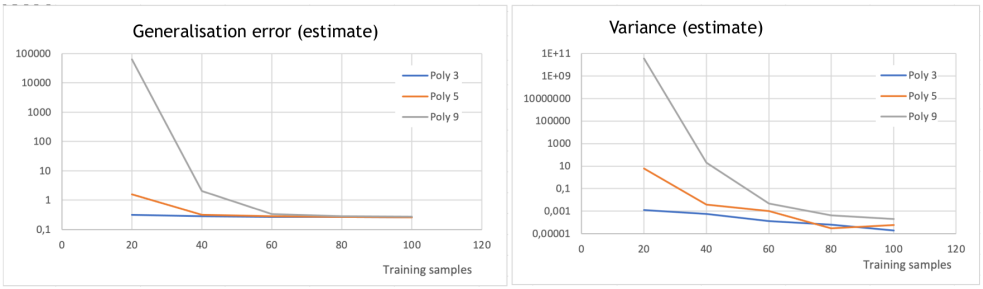


Figure 14: Variation on training dataset size and generalization error with variance with known ground truth and controlled noise; thus, this is an ideal case. Image from lecture 3.

From this, it is possible to observe that, for greater datasets, the generalization error decreases and converges to the irreducible error while the variance converges to zero. It is important to understand that **convergence is slower for more complex models or complex noise**.

LEARNING CURVES: when ground-truth is not available – sampling impact from training dataset
In practice, bias and variance cannot be estimated since one cannot access the underlying data distribution. For this, it is possible to obtain the learning curves that yield information on whether the model has high variance and/or bias by sampling the impact of training dataset size from the total training set. Figure 15 shows the learning curves for two situations.

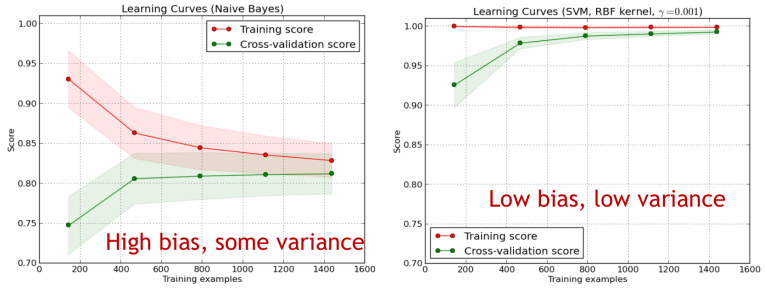


Figure 15: Learning curves with accuracy score; finding learning curves with the loss instead of the score is also possible. On the left-hand side (low N, both plots): insufficient data, severe overfitting, training performance close to perfect, validation performance very bad. On the right-hand side (high N, both plots): hopefully enough data and less overfitting. Limit to infinity: training and validation curves converge to the same value. Image from lecture 3.

Here are a couple of observations:

- If training and validation sets are identically distributed, **and** validation set is large enough: scores should converge to the same value.
- Extrapolation towards *converged value* estimates the best possible generalization error, i.e. model bias + irreducible error.
- More complex models should give lower errors: lower model bias, same irreducible; if not: irreducible error dominates (or other data problem).
- Variability of scores does NOT reflect variance.

Learning vs (Cross)-Validation Curves

- Learning:
  - Plot shows the average training loss/score in the y-axis.
  - Number of samples per training in the x-axis.
- Cross-Validation:
  - Plot shows the average validation loss/score in the y-axis.
  - Serves as an estimate for loss on unseen data.
  - Hyperparameter variation in the x-axis (e.g., regularisation parameter λ, number of features, polynomial degree, training time (#batches) in gradient descent, number of neurons in a neural network, etc.)

3.3 Risks of Data Splitting and IID Assumption Fulfillment

There are some issues when one performs data splitting for model development; for instance, if the training set is too small, there is a risk of overfitting. If the validation set is too small, poor hyperparameter choice is possible. If the test set is too small, the test score might not represent its generalization capacity well. Nevertheless, these are not the only risks of data splitting; one must ensure that the IID assumption is fulfilled as best as possible.

As mentioned, **non-independent data can lead to leakage** of information from validation or test sets into training, resulting in optimistic scores but disappointing performance in the real world. Here are some of the causes:

- Validation and/or test data not sampled independently from train data (the split was not independent). Think of an example!
- Some model or preprocessing parameters are tuned on validation and/or test data. Example: normalizing data based on all data's average and standard deviation (instead of just training data).
- Not enough data to average on latent variables and give identically distributed sets to the three splits.

Example: suppose you have 440 data points from 8 different hospitals; for splitting this dataset, in practice, one has two options

- one randomly shuffles all data, then splits into 3 sets.
  - Positive aspect: three datasets come from the sale distribution; hence, one IID was fulfilled w.r.t. local dataset.
  - Negative aspects: they are not identically w.r.t. the underlying distribution or ground truth; hence, there is a risk of overfitting on these eight hospitals. Furthermore, there is a risk of leakage: records from a single patient could end in training and test sets.
- one randomly selects hospitals to create the three datasets (e.g. 3 for the train, 2 for the validation, and 3 for the test) – this is a **grouped split**.
  - Positive aspect: there is no risk of leakage as **they are independent datasets** (one IID fulfilled w.r.t. underlying distribution).
  - Negative aspect: sets not identically distributed w.r.t. local dataset and underlying distribution.

Mention the other splitting forms and when they are used for specific cases.

3.4 Robust Validation Strategies

When dealing with too little data, it is possible to make use of cross-validation, which consists in averaging across results for multiple train-validate splits given a single dataset. The idea is that one trains the model multiple times (with different train/validation split each time) to then average the validation score across the training runs for model selection. Here are some of the benefits:

- more robust estimate of the error on unseen data,
- allows reducing the size of the validation set (more data for training),
- allows a more in-depth analysis of distribution issues, and
- shows variability across validation errors that combine different training sets (variance) and different validation sets.

Here are some of the types of cross-validation:

- **k-fold cross-validation**: split training data into k folds (typically 5 to 10) considering the IID assumption.

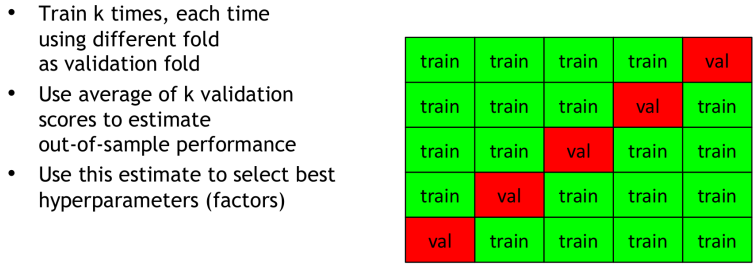


Figure 16: Image from lecture 3.

After final parameter selection: use all train & validation data to train final model – final evaluation on the test set (stage 2 – refit option = True)!

- **Grouped kCV**: assures independence (one of IID) by splitting per group. The number of folds is upper-bounded by the number of groups in the dataset.
- **Stratified kCV**: assures identically distributed (one of IID) by splitting with the assurance of the same amounts of classes in all splits. The number of folds upper-bounded by the number of samples in the smallest class.
- **Stratified-Grouped kCV**: assures both identically distributed and independence (two of IID) by assuring the previous two characteristics.
- Explain multiple random splits and bootstrapping.

The same loss function (for training) and error function (for hyperparameter turning) should be used in all cases. Here are some characteristics according to the number of folds:

- **Fewer folds**: less averaging (more sensitive to the variability of the training set), BUT each fold is more representative because there is more data. This is okay if the given dataset is large. Also, this implies less training time.
- **More folds**: more robust for small datasets and optimally use data for training; however, it takes more time for training.
  - EXTREME CASE: leave-one-out: only one data point is left for validation, and everything else is used for training. **Use LOOCV for small datasets or when estimated model performance is critical!**

3.5 Analysis of Model Performance

When reporting the model scores (train, validation, and test), it is natural to observe two types of gaps: train-validation and validation-test; how large these gaps are will tell the user the possible problems within the model and suggest possible actions to mitigate them. If

- **train-validation gap is too large**: this may be an overfitting issue; here, it is recommended to plot the validation curves and check whether the hyperparameter tuning was correctly done. It is also possible that there are too many features, and feature engineering might be required. Finally, one can also try a simpler model. **If overfitting is not the issue**, it is possible that the corresponding datasets are not identically distributed; for this, plotting learning curves and fold analysis is recommended. If possible, try to increment the size of the validation dataset. The following image shows how to determine whether this is an overfitting or IID issue.

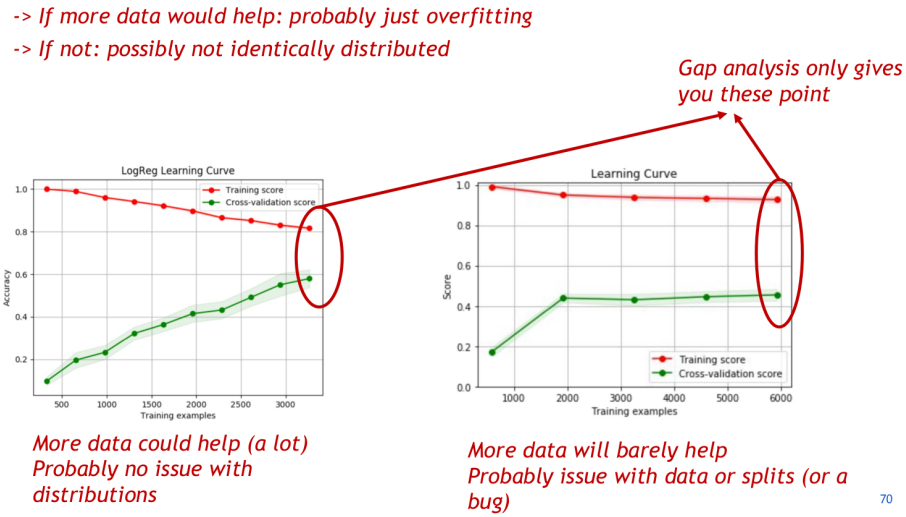


Figure 17: Image from lecture 3.

Explain what exactly fold analysis means.

- **validation-test gap is too large**: leakage may be the main cause of this gap; for this, it is recommended to enhance the splitting of the datasets to ensure independence. If splitting is not the source of leakage, then it is possible that it comes from incorrect preprocessing: everything computed from data MUST be refitted for each fold. It is possible that overfitting in the validation set is the cause due to extreme hyperparameter tunning; for this, the tunning of fewer hyperparameters could help. If possible, adding more data to the validation set is recommended. A violation of the identically distributed can also be the cause; maybe test data was collected separately.
- In both scenarios, falling back on domain knowledge and thinking is recommended.

Gaps indicate unreliable estimation of the algorithm's performance on unseen data; if they remain, it is recommended to attempt a simpler model: higher bias and lower variance can avoid 'unpleasant surprises' in the real world. It is also possible to implement **error analysis**, which requires the implementation of diverse confusion matrices to gain a complete overview of what could be wrong. Mention and explain the three different confusion matrices.

Explain which other types of examinations could be implemented on the given dataset.

3.6 Augmentation

Explain augmentation: why is it needed? How does it differentiate from data cleaning? advantages and disadvantages. How does/can each affect your model? What happens if you do too much or too little of either?

3.7 Summary

Chapter 11: gaps, learning curves, error analysis, model debugging, and augmentation.

Keywords: estimator, model bias and model variance, learning curves, leakage, grouped split, validation strategy,

4 Clustering – K-Means and Clustering Mixture Models

There are instances where data presents specific clusters with some features. It is possible to characterize these clusters to obtain their intra- and inter-cluster distances. With this approach, one can obtain the correlation of random variables, compress the data, or detect anomalies.

4.1 Latent Variables

Latent variables are those that cannot be observed but are relevant to the problem; thus, they are also referred to as hidden variables. These variables can appear naturally (for instance, because of faulty sensors) or be introduced intentionally to model complex dependencies, as shown in the following image.

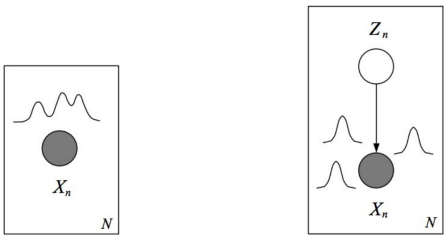


Figure 18: Image from lecture 4.

When latent variables deal with categorical data, it is a clustering problem – which is finding the missing data labels. Otherwise, it is a dimensionality reduction issue – which is finding the missing data inputs.

When there are latent variables present, the algorithm's learning because harder. Why? In a fully observed IID scenario, the probability of a model is just the product of the associated distributions (joint distribution); hence, the logarithm of the likelihood decouples the terms of the associated observed variables.

$$\begin{aligned}\ell(\theta;\mathcal{D}) &= \log p(\mathbf{x},\mathbf{z}|\theta)\\ &= \log p(\mathbf{z}|\theta_z) + \log p(\mathbf{x}|\mathbf{z},\theta_x)\end{aligned}$$

Figure 19: Image from lecture 4.

With hidden variables, the likelihood depends on a sum from the marginalized latent variables, so the log-likelihood still has all parameters coupled together

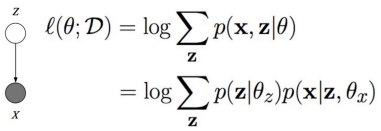


Figure 20: Image from lecture 4.

Not only is learning harder but also this is usually perceived as a black box if latent variables do not have an intuitive interpretation. Furthermore, statistically, hard to guarantee convergence to a correct model with more data. Finally, it usually is more computationally expensive due to its non-closed form for Maximum Likelihood estimates; however, the Expectation-Maximization (EM) algorithm provides a general-purpose local search algorithm for learning parameters in probabilistic models with latent variables.

Nevertheless, not everything is bad when latent variables are present in a situation, as their presence can enhance generalization with their applications; some are learning with missing data, discovering new features, and more.

Professor Q.: *Discuss latent variables in the continuous case vs discrete case, and discuss pros/cons of latent variables.*
Professor Q.: *How to deal with unobserved/latent variables?*

4.2 K-Means Clustering

Intuitively, a cluster is a group of points close together and far from others.

Professor Qs.:*Explain the K-Means clustering algorithm. Is (global) convergence guaranteed? What is the computational complexity of K-Means?*

For the following algorithm, annotate the computational complexity per step.

- **K-Means Clustering**
  - **Initialize** K centroids (cluster means), then **iterate** between two phases:
    - **step 1** (E): given centers, **assign points** to nearest "cluster"
    - **step 2** (M): given points, **update centers** to be the new cluster means

Write and explain the cost function of this problem.

Because KMeans is a (Euclidean) distance-based algorithm, it is strongly suggested to first rescale the data. Make notes to the following image indicating the steps from the algorithm.

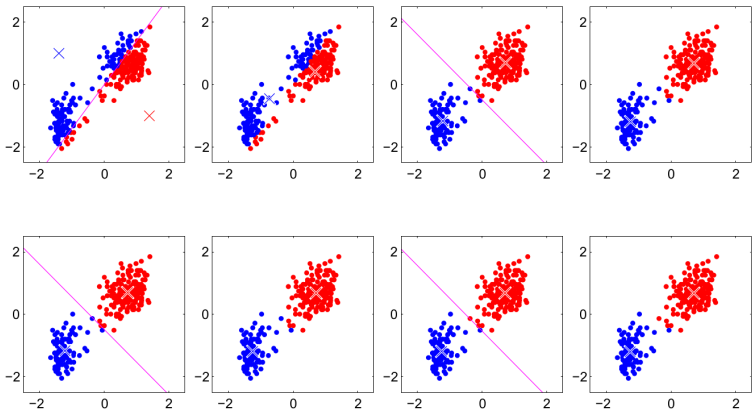


Figure 21: Image from lecture 4.

The algorithm is **guaranteed to converge** after a finite number of iterations; if it converges in a local minimum or a global minimum is strongly dependent on the initialization of the K centroids.

Professor Q.: *What is vector quantization?*

One of the common applications of KMeans is Image Compression (also referred to as vector quantization). For instance, in the image below, the results vary after compressing the image with K colors.

Each pixel (color value) is one data point in RGB space; considering this, all pixels in a cluster can be colored by the cluster's mean. This is known as hard assignments of data points to clusters.

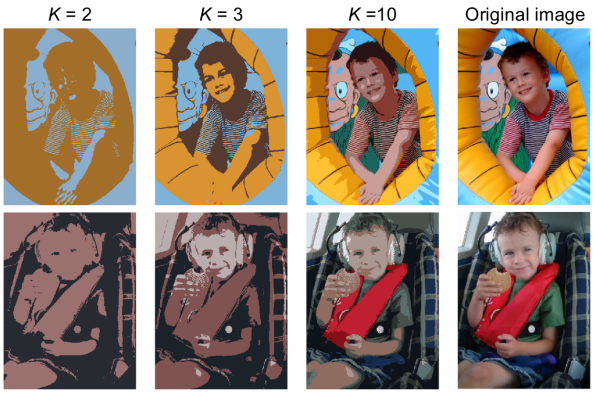


Figure 22: Image from lecture 4.

Professor Q.: *How to select the optimal number "K"?*

The problem at hand can determine the selection of the best K. For instance, if there is a meaning to K, it is possible to manually set this value. When the meaning is not relevant, one can make use of the **incremental (leader-cluster) algorithm**, which consists in adding one-at-a-time "K" until *elbow* is shown in the objective (error) function; see the following image.

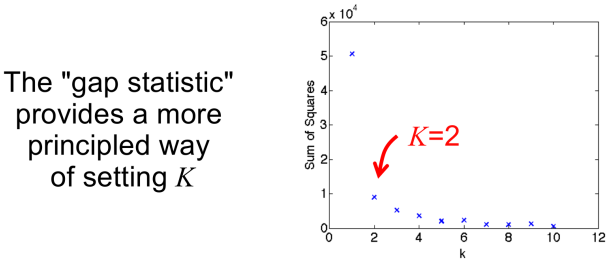


Figure 23: Image from lecture 4.

Professor Q.: *Discuss pros/problem cases/extensions of K-means.*

The positive aspect of this algorithm is that it is fast and converges to a local minimum of within-cluster squared error. However, it is extremely sensitive to the initialization of the centroids and outliers; furthermore, it can only detect spherical/circular clusters.

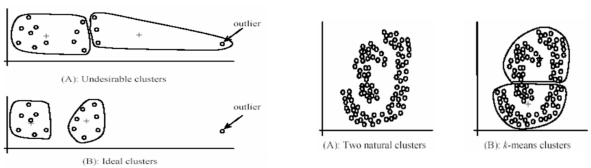


Figure 24: Image from lecture 4.

For extensions, there are

- Speed-ups: possible through efficient search structures, and
- General distance measures: k-medoids.

4.3 Mixture Distributions: Mixture of Gaussians

The Mixture of Gaussians algorithm attempts to approximate an observed smooth distribution as a sum of M individual latent normal distributions. Additive mixture models are the most basic possible latent variable model, having one single discrete latent variable.

Write the associated mathematics to the generative model of GMM, explain each parameter, and draw the graphical model.

Professor Qs.: *Explain the key differences between MoG and K-Means clustering. Discuss MoG properties and problems. Draw and explain a MoG generative model as a directed graphical model. Explain all variables.*

Professor Q.: *Does the MoG model has (a) discrete and/or continuous latent variable(s)? Explain.*

Professor Q.: *Model selection problem: How to select the optimal number of mixture components ('K') for a MoG model?*

Professor Q.: *Is a closed-form analytical Maximum Likelihood estimation (MLE) possible for a MoG model? Explain.* Do the mathematics.

4.4 Expectation-Maximization (EM) Algorithm

Explain the mathematics of the second approach and the need for an estimator j.

Professor Q.: *Discuss the EM algorithm for MoG: discuss (1) the "Credit Assignment Problem" and (2) the "Mixture Density Estimation."*

Professor Q.: *Is regularization needed in EM? Why? How? In practice, how to calculate Mixture of Gaussians? Give tips/tricks (technical advice).*

4.5 Summary

5 The Data Pipeline (DP) and Ramp-Up Towards Non-Linear Models

When having categorical data, one part of the data pipeline is to convert them to numerical values by using *OrdinalEncoders* (1, 2, 3, ...) or *OneHotEncoders* (0,1; 00, 01, 10, 11; 100, 010, 001, ...). A general pipeline is of the following form

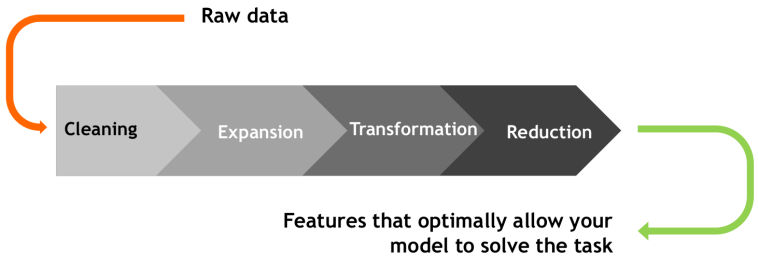


Figure 25: Image from lecture 5.

Figure 26: Image from lecture 5.

Sometimes, augmentation might be required at the beginning to increase the size. Remember that many steps in the data-pipeline use parameters that are extracted from the whole training set, these MUST be included in your sklearn cross-validation pipeline to avoid leakage!

5.1 DP: Cleaning

Sometimes, with outliers, missing or zero values, action must be taken for optimal algorithm performance; for instance, filling missing values with the mean of a specific group and/or date-time. The data cleaning will always depend on the specific problem at hand. Here are some options implemented by sklearn.

sklearn.impute: Impute

Transformers for missing value imputation

User guide: See the Imputation of missing values section for further details.

impute.SimpleImputer(* [, missing_values, ...])	Imputation transformer for completing missing values.
impute.IterativeImputer([estimator, ...])	Multivariate imputer that estimates each feature from all the others.
impute.MissingIndicator(* [, missing_values, ...])	Binary indicators for missing values.
impute.KNNImputer(* [, missing_values, ...])	Imputation for completing missing values using k-Nearest Neighbors.

Figure 27: Image from lecture 5.

While outliers can damage the algorithm's performance, sometimes it is not correct to remove them from the data; sometimes, the correct approach is to keep them and aim for a more robust algorithm. This depends strongly on the

problem at hand. The cleaning part is the opposite of augmentation, as the former aims to ease the complexity of the data while the latter increases it.

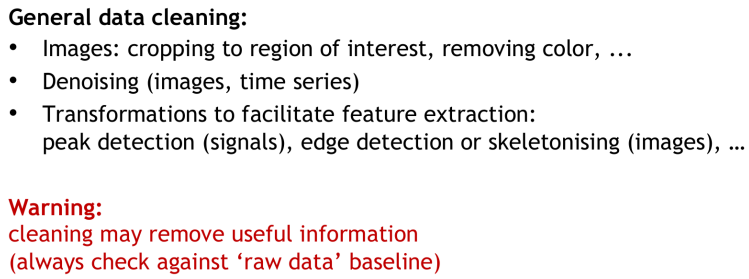


Figure 28: Image from lecture 5.

Data cleaning = removing (unnecessary) information. Data augmentation = adding (necessary) information.

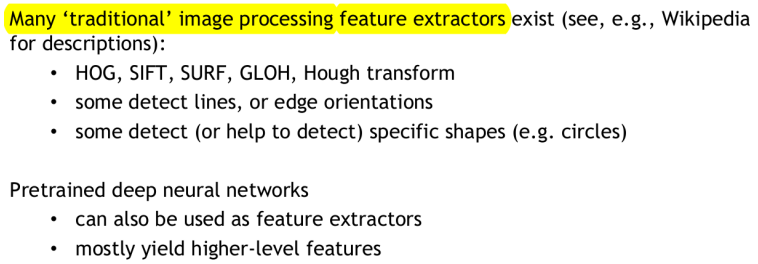
Professor Q.: *What is (or can be) the impact of outliers on your model? What are ways to deal with outliers? Give examples of situations where you would make different choices.*

Professor Q.: *Preprocessing: what is "standardization" or "feature normalization" in a machine learning pipeline? Why/when do you need it? What is the difference between sklearn "scalers", "normalizers" and power transformers? When would you use which?*

5.2 DP: Feature Extraction / Expansion

Think: how does it compare to cleaning vs preprocessing vs transformations?

When features are not good enough for a model, sometimes it is required that they get transformed (non-linearly or linearly, e.g. polynomials) to obtain their useful characteristics. It is good to remember that many models do not like sparse features. Unsupervised learning can also be utilized to extract new features.



5.3 DP: Transformations and Embeddings

"An embedding is a relatively low-dimensional space into which you can translate high-dimensional vectors. Embeddings make it easier to do machine learning on large inputs like sparse vectors representing words." [13]

Professor Q.: *What is feature "orthogonalization"? How/why can it help? What is the difference between orthogonalization and dimensionality reduction?*

5.3.1 Principal Component Analysis: unsupervised

5.3.2 Linear Discriminant Analysis: supervised

LDA is a supervised generative model that allows computing the class probabilities using linear discriminants. Since probabilities add up to 1: only k-1 discriminants for k classes

5.4 DP: Dimensionality Reduction

Professor Q.: *What is dimensionality reduction? What is its purpose? What are the trade-offs you need to make? Dimensionality reduction: unsupervised vs. supervised techniques, "kbest" subset selection, recursive feature addition/removal,...*

Most important: simpler models are more robust to overfitting (on too small datasets) – lower variance! It also reduces the computational time in improves inference (curse of dimensionality).

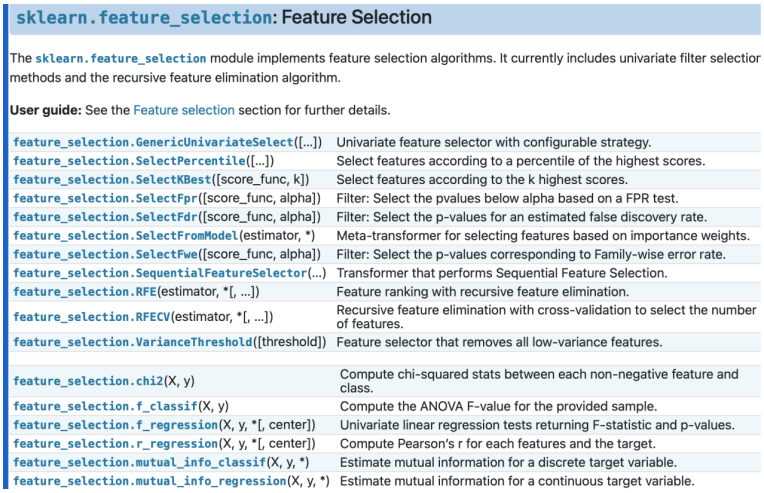
Dimensionality reduction means considering a subset of all the available features or using a 'compressive' transformation (e.g. LDA or PCA + cut-off).

Typical workflow:

- first consider many features,
- transform them to compress information,
- select the best ones (subset selection)

IMPORTANT: there are 2^F subsets of F features.

To study features, you can add the best features one by one (forward analysis) or start with all and remove them one by one (backward analysis). One can also use **feature importance** by permutations; other methods could be implemented like L_1 regularization, and so on. There are many methods to investigate feature importance and effectively reduce dimensionality. Sklearn offers many built-in methods.



sklearn.feature_selection: Feature Selection

The sklearn.feature_selection module implements feature selection algorithms. It currently includes univariate filter selectior methods and the recursive feature elimination algorithm.

User guide: See the Feature selection section for further details.

feature_selection.GenericUnivariateSelect([...])	Univariate feature selector with configurable strategy.
feature_selection.SelectPercentile([...])	Select features according to a percentile of the highest scores.
feature_selection.SelectKBest([score_func, k])	Select features according to the k highest scores.
feature_selection.SelectFpr([score_func, alpha])	Filter: Select the pvalues below alpha based on a FPR test.
feature_selection.SelectFdr([score_func, alpha])	Filter: Select the p-values for an estimated false discovery rate.
feature_selection.SelectFromModel(estimator, *)	Meta-transformer for selecting features based on importance weights.
feature_selection.SelectFwe([score_func, alpha])	Filter: Select the p-values corresponding to Family-wise error rate.
feature_selection.SequentialFeatureSelector(...)	Transformer that performs Sequential Feature Selection.
feature_selection.RFE(estimator, *[, ...])	Feature ranking with recursive feature elimination.
feature_selection.RFECV(estimator, *[, ...])	Recursive feature elimination with cross-validation to select the number of features.
feature_selection.VarianceThreshold([threshold])	Feature selector that removes all low-variance features.
feature_selection.chi2(X, y)	Compute chi-squared stats between each non-negative feature and class.
feature_selection.f_classif(X, y)	Compute the ANOVA F-value for the provided sample.
feature_selection.f_regression(X, y, *[, center])	Univariate linear regression tests returning F-statistic and p-values.
feature_selection.r_regression(X, y, *[, center])	Compute Pearson's r for each features and the target.
feature_selection.mutual_info_classif(X, y, *)	Estimate mutual information for a discrete target variable.
feature_selection.mutual_info_regression(X, y, *)	Estimate mutual information for a continuous target variable.

Figure 29: Image from lecture 5.

5.5 Introduction to Kernels

As already discussed, it is possible that a non-linear boundary exists for classified data in a given dimension and a linear boundary for the same dataset for another dimension, as shown in the following example.

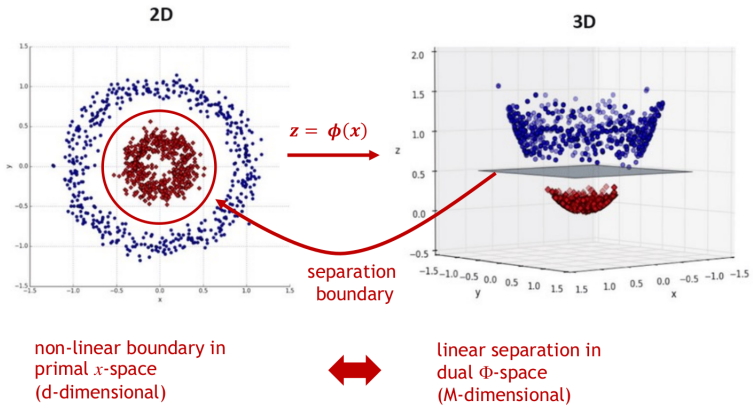


Figure 30: Image from lecture 5.

Here, the primal space d-dimensional space of the original data $\mathbf{x}$ (d is the number of features). The dual space is the M-dimensional space of $\Phi(\mathbf{x})$ where the linear separation exists. One can perform regression or classification in either the primal or dual space.

In several machine learning algorithms, one can find the explicit expression of the inner product between the training examples and the new data when computing $\hat{y}(\mathbf{x'})$. If the tasks require the algorithm's learning on the transformed features, then one would need to compute the inner product of transformed feature vectors $\Phi(\mathbf{x})^T\Phi(\mathbf{x'})$; here is where the concept of kernels comes in hand.

A kernel $K(\mathbf{x}, \mathbf{x'})$ is an inner product between two vectors in some vector space in which the vectors $\Phi(\mathbf{x})$ are transformed versions of $\mathbf{x}$. The only requirement is that the kernel can be expressed as an inner product; for instance

$$K(\mathbf{x}, \mathbf{x'}) = \Phi(\mathbf{x})^T\Phi(\mathbf{x'}) = \sum_{j=1}^{M} \phi_j(\mathbf{x})\phi_j(\mathbf{x'}) \tag{14}$$

which implies that it is always a symmetric function of two d-dimensional vectors; hence, its result is a scalar, not a vector. Have a look at *the kernel trick* for linear regression on transformed features

$$y(\boldsymbol{x'}) = \sum_{t=1}^{N} \alpha_t \; r_t\big(\boldsymbol{\phi}(\boldsymbol{x}_t)^T\boldsymbol{\phi}(\boldsymbol{x'})\big) + w_0 \; = \sum_{t=1}^{N} \alpha_t \; r_t \, \mathbf{K}(\boldsymbol{x}_t, \boldsymbol{x'}) + w_0$$

By rewriting a linear model in inner product form, it no longer depends on $\boldsymbol{\phi}(\boldsymbol{x'})$, but only on the kernels

If we know the kernel, we don't need to know the feature space explicitly, ***we only need to know it exists!***

Figure 31: Image from lecture 5.

This means there is no need to compute the dual space for effective learning! This property makes inference faster and better; for instance, polynomial regression is more effective with a q-polynomial kernel $K(\mathbf{x}, \mathbf{x'}) = (\mathbf{x}^T\mathbf{x'} + 1)^q$ because computing the kernel function is less costly than computing the transformation and the inner product.

In short: a positive definite symmetric function is a kernel, meaning: $k(v, w) = k(w, v)$.

Let X be a non-empty set. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called positive definite kernel function, iff
k is symmetric, i.e. $k(x, x') = k(x', x)$ for all $x, x' \in \mathcal{X}$, and
for any set of points $x_1, \ldots, x_n \in \mathcal{X}$, the matrix

$$K_{ij} = (k(x_i, x_j))_{i,j}$$

is positive (semi-)definite, i.e. for all vectors $\mathbf{x} \in \mathbb{R}^n$:

$$\sum_{i,j=1}^{N} \mathbf{x}_i K_{ij} \mathbf{x}_j \geq 0$$

Figure 32: Image from lecture 5.

Give examples of other kernels and their properties.

Explain how comparing angles is faster with kernels than its computation in dual space.

Give memory storage of kernels and dual space and mention the main drawbacks of its implications.

Kernels can be used to introduce non-linearity whenever **a model only depends on the data through the inner products** between the new feature vector and the training feature vectors.

Mention the main difference between KNN, SVM, and kernel SVM.

5.6 Summary

6 Directed and Undirected Graphical Models

Professor Q.: *Describe the key properties of graphical models. Why useful?*

This lecture outline:

- **Generative Models**: model the likelihood and prior distributions separately.
  - Bayesian Networks
  - Markov Random Fields
  - Exact Inference
- **Discriminative Approaches**: model the posterior distribution
  - Ensemble methods and boosting
  - Random Forests and Decision Trees

To understand the content of this lecture, it is necessary to review the requirements in Probabilistic Models.

A probability is an expression of belief about an uncertain event; this proves useful for decision-making and combining sources of information. A key quantity in probabilistic reasoning is the **joint distribution** $p(x_1, \ldots, x_k)$ of k random variables in a model. It is possible to classify machine learning tasks in terms of the joint distribution:

- **inference**: given the joint distribution, queries are done. When considering a boolean case with k entries, then summations take $\mathcal{O}(2^k)$ and inference is slow for a large k;
- **learning**: modeling the joint distribution (requires inference). For the same boolean case, it implies that 2^k parameters must be learned.

Both require the manipulation of the joint distribution. Here are the main rules of probability that must be kept in mind:

- **Sum rule**: is the marginalization over discrete or continuous random variables

$$p(X) = \sum_y p(X, Y = y); \quad \text{or} \quad p(X) = \int p(X, Y = y)dy \tag{15}$$

- **Product rule**:

$$p(X, Y) = p(X)p(Y \mid X) \tag{16}$$

- **Bayes rule**:

$$p(Y \mid X) = \frac{p(X \mid Y)p(Y)}{p(X)} \tag{17}$$

- **Conditional Independence**: X and Y are independent iff
  - $p(X \mid Y) = p(X)$ and $p(Y \mid X) = p(Y)$, or
  - $p(X, Y) = p(X)p(Y)$

  In many cases, the use of conditional independence greatly reduces the size of the representation of the joint distribution, which directly reduces the number of parameters to be learned in inference or increases the computational speed in inference.

6.1 Introduction to Graphical Models

Professor Q.: *What is "Causal inference"? What is "Diagnostic inference"?*

Graphical models characterize for merging Probability Theory with Graph Theory, providing a natural tool for uncertainty and complexity problems by giving a graphical representation of probability distributions.

Graphical models are composed by two elements:

- **nodes** (also called vertices) represent observed or unobserved**random variables** or groups of random variables;

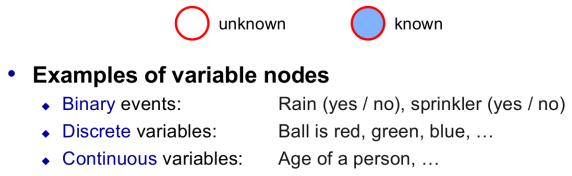


Figure 33: Image from lecture 6.

and,

- **edges** (also called links) represent **conditioning**; they can be directed (known as Bayesian Networks) or undirected (Markov Random Fields).

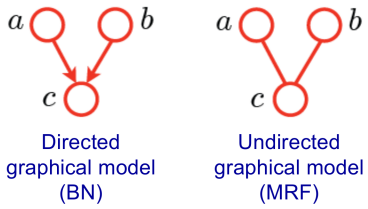


Figure 34: Image from lecture 6.

6.2 Directed Graphical Models: Bayesian Networks

Bayesian networks are based on a directed acyclic graph (DAG); however, they can be fully connected, known as associated graphs. Because this type of model links are directed, they represent causal dependencies among the (observed or unobserved) random variables. The structure of the network qualitatively describes the dependencies of the random variables.

Professor Q.: *How to get a factorized representation of the joint distribution in directed graphical models?*

Joint Probability in Directed Graphs
Consider the node A directed to node B; this means that the value of B depends on A; this dependency is expressed as a conditional probability $p(B \mid A)$; and the probability of A is given by the prior probability $p(A)$. **The complete graphical model describes the joint probability distribution** $p(A, B) = p(A)p(B \mid A)$. Given the joint probability, it is possible to derive the marginalization of A in B and obtain the other conditional distribution; namely,

$$p(B) = \sum_A p(A, B) = \sum_A p(B|A)p(A); \quad \text{and} \quad p(A \mid B) = \frac{p(A, B)}{p(B)}, \tag{18}$$

with the latter being Baye's rule.

Factorization in Directed Graphs
Consider a Bayesian Network consisting of a set of variables $U = x_1, \ldots, x_n$ forming an acyclic-directed graph. The joint distribution of said model can be expressed as

$$p(x_1 \ldots, x_n) = \prod_{i=1}^{n} p(x_i \mid \{x_j \mid j \in \text{pa}_i\}), \tag{19}$$

with pa_i denoting the parent nodes of x_i, a factorized representation of the joint. In other words, one can express the joint as a product of all the conditional distributions from the parent-child relations in the graph.

Exercise: Computing the joint probability

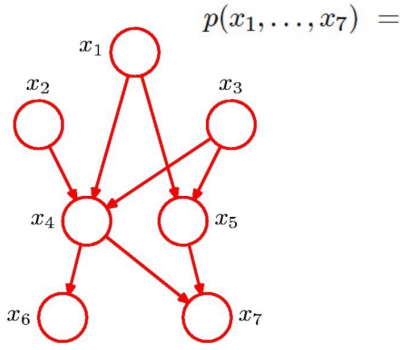


Figure 35: Image from lecture 6.

Overall, having the factorized version of the joint probability allows reducing the complexity of the problem from $\mathcal{O}(2^n)$ to $\mathcal{O}(n \cdot 2^k)$, with k denoting the maximum number of parents of a node in the model. Here is an example of the Intensive Care Unit's alarm network for monitoring patients.

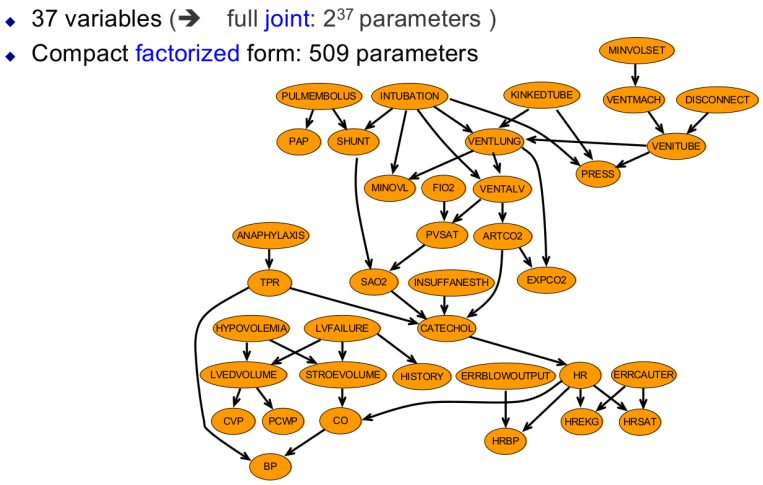


Figure 36: Image from lecture 6.

Professor Q.: *Discuss "conditional independence" in directed graphical models. Discuss the Divergent, Chain and Convergent cases.*

Conditional Independence
The notion of conditional independence means that other variables become independent given the observation of a certain variable; e.g. $p(x_2 \mid x_0, x\ 2) = p(x_2 \mid_x 1)$, means that x_2 is conditionally independent from x_0 when x_1 is observed. The nomenclature for conditional independence is the following

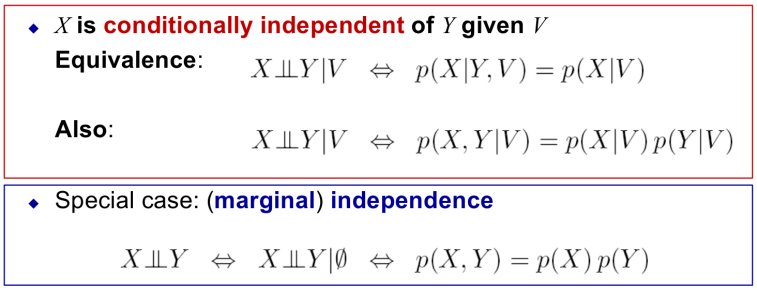


Figure 37: Image from lecture 6.

In other words, by observing V, the observation of Y is irrelevant for the estimation of X, and vice versa. Furthermore, knowing V explains any observed dependence between X and Y.

It is possible to read the conditional independence from directed graphical models; it is typical to encounter three canonical cases, each composed of three nodes (A, B, C):

- **Tail-to-Tail** (divergent): this model has C as the only parent of A and B ($A \leftarrow C \rightarrow B$). To tell whether A and B are independent, one needs to marginalize out C; namely,

$$\begin{aligned} p(A,B) &= \sum_C p(A,B,C) \\ &= \sum_C p(A \mid C)p(B \mid C)p(C) \\ &\neq p(A)p(B) \rightarrow A \not\perp\!\!\!\perp B \mid \emptyset \end{aligned} \tag{20}$$

  If C is not observed, then one can express the conditional probability as

$$p(A,B \mid C) = \frac{p(A,B,C)}{p(C)} = \frac{p(A \mid C)p(B \mid C)p(C)}{p(C)} = p(A \mid C)p(B \mid C) \rightarrow A \perp\!\!\!\perp B \mid C \tag{21}$$

  The latter is because there is no link between B and A; with the former equation, one observes that A and B are conditionally independent!

- **Head-to-Tail** (chain): this model has A as the unique partent of C and C the unique parent of B ($A \rightarrow C \rightarrow B$). Again, are A and B generally independent?

$$\begin{aligned} p(A,B) &= \sum_C p(A,B,C) \\ &= \sum_C p(A)p(C \mid A)p(B \mid C) \\ &\neq p(A)p(B) \rightarrow A \not\perp\!\!\!\perp B \mid \emptyset \end{aligned} \tag{22}$$

What happens if C is observed?

$$p(A,B \mid C) = \frac{p(A,B,C)}{p(C)} = \frac{p(A)p(C \mid A)p(B \mid C)}{p(C)}, \text{ with } p(C \mid A) = \frac{p(A \mid C)p(C)}{p(A)} \rightarrow A \perp\!\!\!\perp B \mid C \tag{23}$$

it is possible to observe that A and B are conditionally independent when C is observed.

- **Head-to-Head** (Convergent): this model has A and B being the two parents for C ($A \rightarrow C \leftarrow B$). Are A and B independent?

$$p(A,B) = \sum_C p(A,B,C) = \sum_C p(A)p(B)p(C \mid A,B) = p(A)p(B) \rightarrow A \perp\!\!\!\perp B \mid \emptyset \tag{24}$$

What happens if C is observed?

$$p(A,B \mid C)\frac{p(A,B,C)}{p(C)} = \frac{p(A)p(B)p(C \mid A,B)}{p(C)} \rightarrow A \not\perp\!\!\!\perp B \mid C \tag{25}$$

Therefore A and B are **not** conditionally independent when C is observed; this also holds when any of C's descendants is observed. This case is the opposite of the previous cases.

Visit the explanation on conditional independence given the example in the lecture (cloudy days).

Explaining Away
Professor Q.: *What is "explaining away"?*

There are instances where observing a child node changes the probability of a parent(s) node(s) but do not block them. **The phenomenon of explaining away means that observations of the child nodes will not block paths to the co-parents.**

In other words, Bayesian Networks have the capability to explain away hypotheses by new evidence.

The Markov Blanket
Professor Q.: *What is a "Markov Blanket" in directed graphical models?*

A Markov Blanket, or Markov boundary, of a node A is the minimal set of nodes that isolates A from the rest of the graph, comprising the set of parents, children, and co-parents (the other parents of its children) of A.

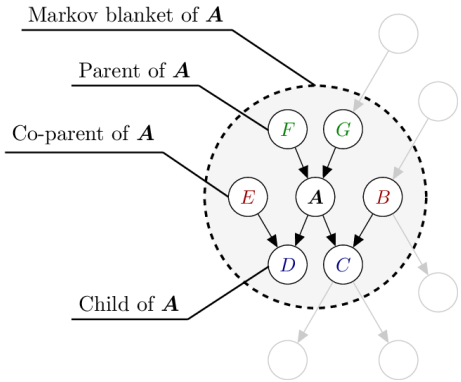


Figure 38: Image from [14].

6.3 Undirected Graphical Models: Markov Random Fields

Markov Random Fields (MRFs) are undirected graphical models which allow easier reading of the conditional independence on their nodes. The Markov Blanket of MRFs are just the neighboring nodes (no need to worry of co-parents).

Professor Q.: *What is a "Markov Blanket" in Undirected graphical models?*

Conditional Independence
If every path from any node in set A to set B passes through at least one node in set C, then $A \perp\!\!\!\perp B \mid C$.

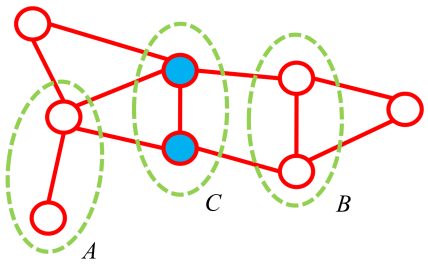


Figure 39: Image from lecture 6.

The analysis just requires simple graph separation. Checking all the paths between sets can tell whether a set its conditional independence. Here is another example between two computers and two hubs.

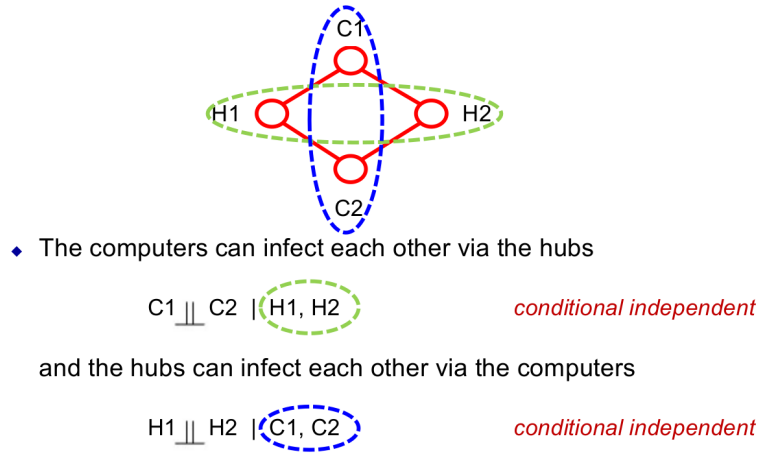


Figure 40: Image from lecture 6.

Professor Qs.: *ow to factorize an undirected graph? What is a (max) clique? Why is a normalization constant needed?*

Factorization in Undirected Graphs
To understand how factorization is done in MRFs, it is important to understand the concept of **clique**. A clique is a fully connected subset of the nodes; a link exists between all pairs of nodes in the subset. The **maximal clique** is the biggest possible such clique in a given graph.

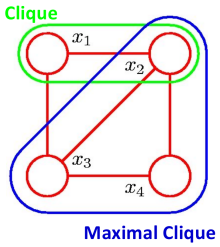


Figure 41: Note that nodes in a clique cannot be made conditionally independent from each other. Image from lecture 6.

The joint distribution of a non-directed graph is expressed as a product of non-negative potential functions $\psi_c(\mathbf{x}_c)$ over maximal cliques in the graph

$$p(\mathbf{x}) = \frac{1}{Z}\prod_c \psi_c(\mathbf{x}_c), \tag{26}$$

with Z being the partition function

$$Z = \sum_{\mathbf{x}}\prod_c \psi_c(\mathbf{x}_c), \tag{27}$$

that normalizes the probability. One of the main limitations of MRFs is the partition function since its evaluation is $\mathcal{O}(K^M)$, where M is the total number of nodes and K is the number of nodes in a given clique (?). Expressing the

potential function as an exponential function is convenient due to their similarity in Statistical Physics for the Boltzmann distribution. Furthermore, it is convenient because the joint distribution depends on the product of these components; hence, taking the logarithm to the resultant expression is suitable.

One can observe that Bayesian Networks are automatically normalized, while MRFs are not.

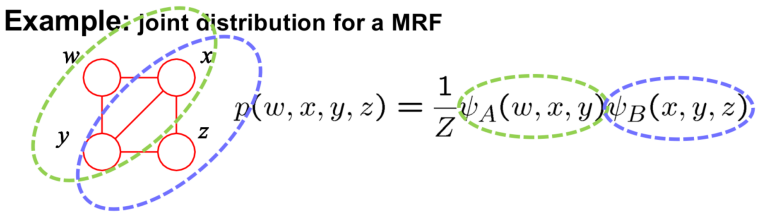


Figure 42: Image from lecture 6.

Example of usage: Ising model for image restoration, segmentation, inpainting, and denoising. MRFs can also be used to convert a low-resolution image into a high-resolution image or create disparity maps.

Professor Q.: *Compare Directed vs. Undirected Graphs: pros/cons*

Professor Qs.: *How to convert a directed to an undirected graph? What is "moralization"? What is a "moral graph"?*

It is possible to convert a directed graph into an undirected graph by considering the necessary cliques to replace the directed links. Furthermore, one must *marry the parents*, or introduce additional links, to account for normalization in the directed graph. This process is known as **moralization**.

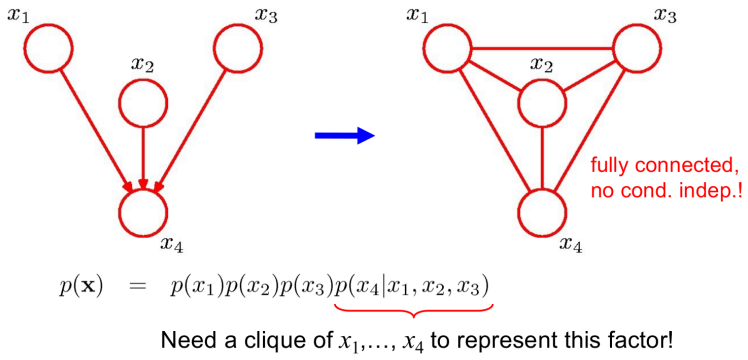


Figure 43: Image from lecture 6.

A **moral graph** is a graph that results from transforming a Bayesian Network into a MRF by moralizing it. Here is the general procedure:

1. Moralize the graph (marry the parents).
2. Drop the arrows of the original graph (obtain a moral graph).
3. Find maximal cliques for each node and initialize all clique potentials to 1.
4. Take each conditional distribution factor of the original directed graph and multiply it into one clique potential.

WARNING: conditional independence properties are often lost and moralization results in additional connections and larger cliques.

EXERCISE: take the Bayesian network in Figure 35 and transform it into a Markov Random Field.

6.4 Exact Inference in Graphical Models

Professor Q.: *What is inference in graphical models (only high-level description: general definition and general question formulation $p(Q \mid E)$: max 10 lines)?* Watch the lecture, if possible.

6.5 Summary

Keywords: associated graphs, Markov blanket, clique, potential function, partition function, moralization

7 Bayesian Estimation

7.1 Recap: Regression

Briefly explain linear regression and Ridge regression (L_2 regularization).

NEXT: Bayesian approach:

Professor Qs.: *How can we prevent overfitting systematically? How can we avoid the need for validation on separate test data?*

7.2 Probability Concepts

Recall the sum rule, product rule, Baye's rule, and marginalization – the basic rules of probability. The following terms must be understood:

- **Prior**: a priori probability; namely, what do we know before seeing the data?
- **Conditional probability**: the probability of observing a random variable given the observation of another one.
- **Interpretation of Bayes Theorem/Rule**:

$$posterior = \frac{likelihood \times prior}{Normalization\ factor} \tag{28}$$

- **One-dimensional Gaussian Distribution**:

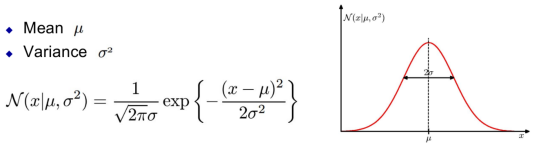


Figure 44: Image from lecture 7.

- **Multi-dimensional Gaussian Distribution**:

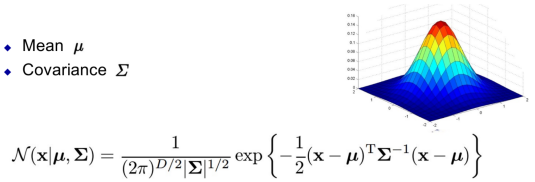


Figure 45: Image from lecture 7.

IMPORTANT: the inverse of the covariance matrix is known as the precision matrix: $\Lambda = \Sigma^{-1}$. Write the terms of the one- and multi-dimensional Gaussian distribution in terms of precision beside their images.

- **Special Cases of Gaussian Distributions**

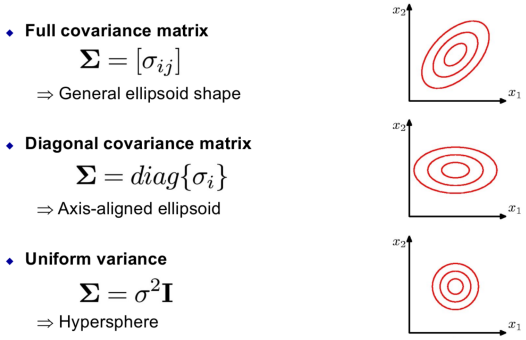


Figure 46: Image from lecture 7.

- **Central Limit Theorem**: The distribution of the sum of N i.i.d. random variables becomes increasingly more Gaussian as N grows.
- **Maximum Likelihood**: estimation of the parameters $\boldsymbol{\theta}$ through the minimization of the negative log-likelihood. Applying maximum likelihood to estimate the parameters of a Gaussian yields

$$\begin{aligned}
\mu_{ML} &= \frac{1}{N}\sum_{i=1}^{N} x_i \quad \text{sample mean} \\
\sigma^2_{ML} &= \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_{ML})^2 \quad \text{(biased) sample variance} \\
\tilde{\sigma}^2_{ML} &= \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_{ML})^2 \quad \text{(unbiased) sample variance}
\end{aligned} \tag{29}$$

One of this approach's limitations is that **it systematically underestimates the variance of the distribution**, resulting in a tendency to overfit the training data.

Professor Qs.: *Does Maximum Likelihood (ML) estimation for the parameters of a Gaussian distribution under or overestimate the true mean and/or variance? Explain + give math expressions. What happens if there is only one sample?*

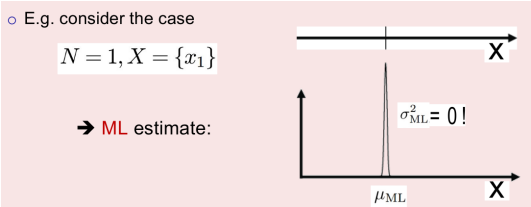


Figure 47: Image from lecture 7.

This method is a Frequentist concept, meaning that probabilities are the frequencies of random, repeatable events – it refers to *past* events; these frequencies are fixed but can be estimated more precisely when more data is available. **This is in contrast to the Bayesian interpretation**. In the Bayesian view, probabilities quantify the uncertainty about certain states or events – it refers to the *future*. This uncertainty (measures of belief) can be revised in the light of new evidence.

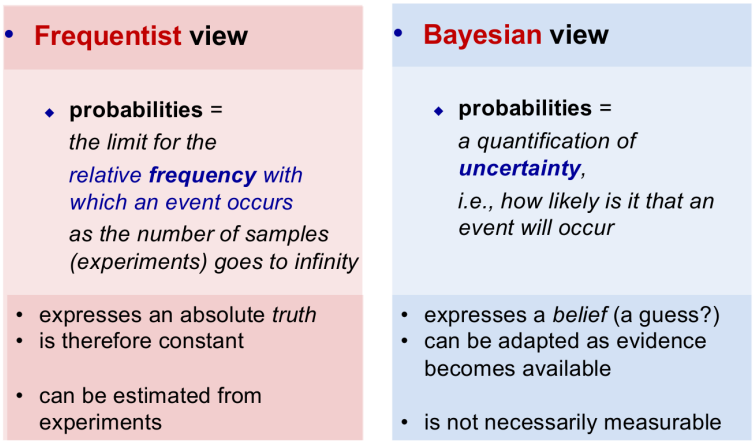


Figure 48: Image from lecture 7.

Bayesian Learning Approach
One of the main differences between Bayesian Learning and Frequentist Learning is that in the former visualizes $\boldsymbol{\theta}$ as a random variable, while the latter perceives it as a fixed parameter that can be estimated. In Bayesian learning, $p(\boldsymbol{\theta})$ is the prior distribution. The probability density of the parameters after the training data is observed $p(\theta \mid x)$ is the posterior probability density.

Bayesian Learning is a Generative Model since the prediction accounts for finding the probability distribution for a new input value x conditioned on the observed training data. It is possible to analyze this without labels, such as

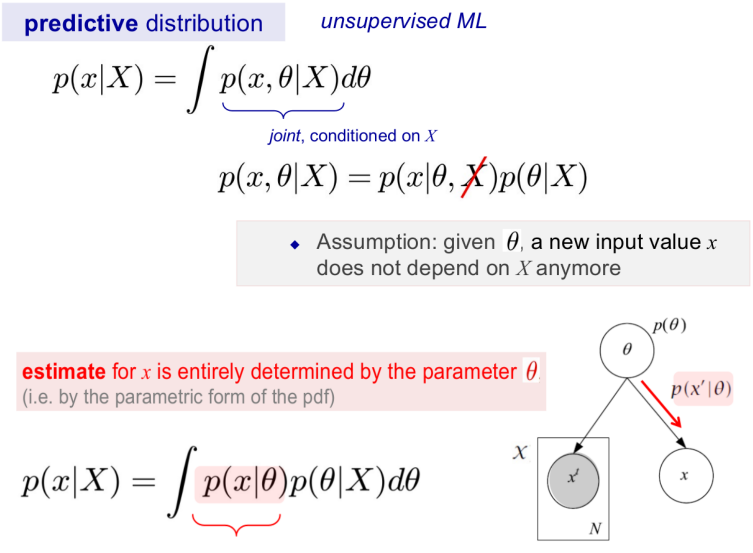


Figure 49: Image from lecture 7.

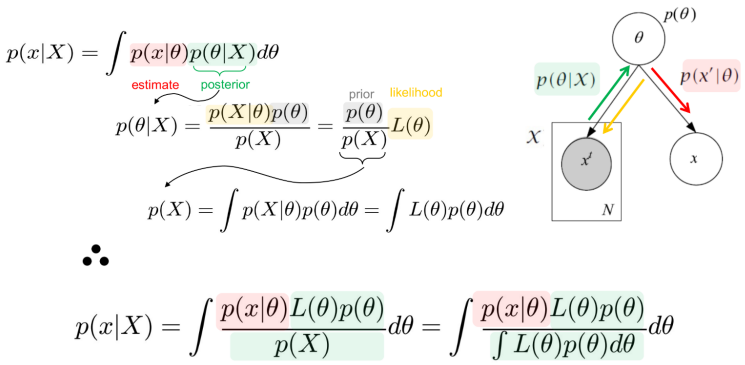


Figure 50: Image from lecture 7.

The wrap-up of Bayesian Learning is that the more uncertain one is about θ, the more one averages over all possible parameter values.

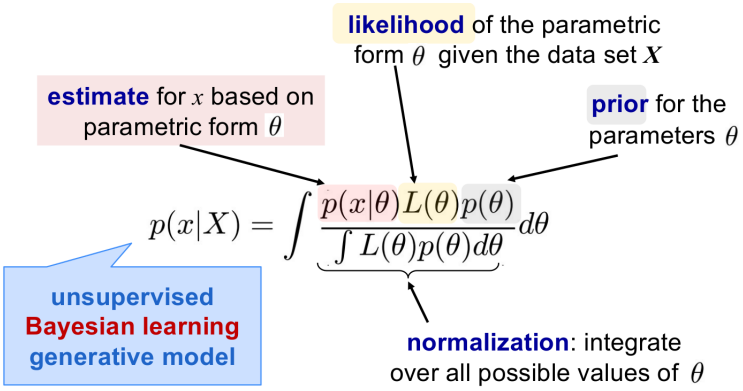


Figure 51: Image from lecture 7.

Professor Q.: *Bayesian learning: Draw and explain a generative model (as a directed graphical model) in the unsupervised case, and give the predictive probability $p(x' \mid x)$ for a new data sample x', conditioned on the observed (training) data x*

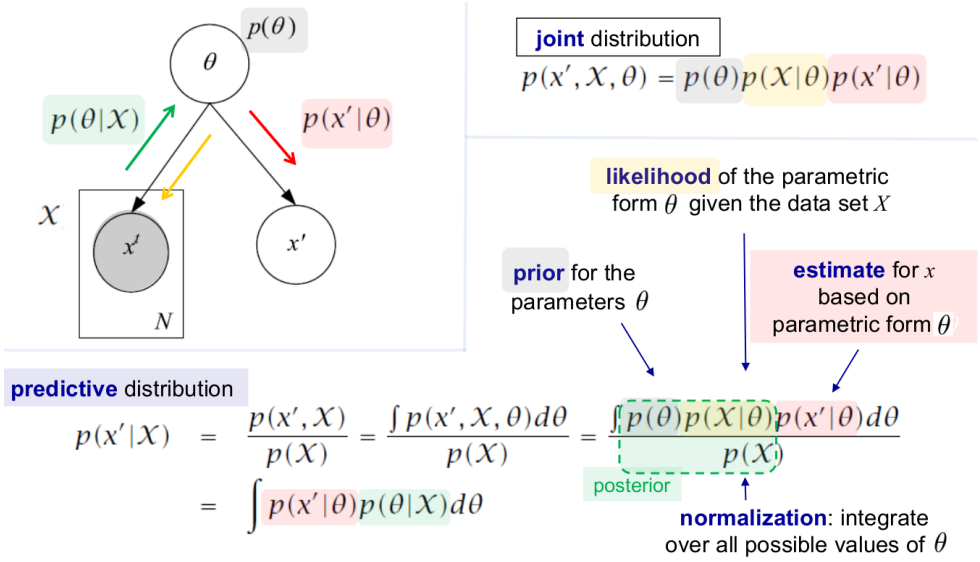


Figure 52: Image from lecture 7.

7.3 A Probabilistic View on Regression

7.3.1 Least-Squares Estimation as Maximum Likelihood

!! **Professor Q.:** *Linear regression: Prove and explain that Least-Squares regression is equivalent to Maximum Likelihood*

(ML) estimation under the assumption of Gaussian output noise. Derive and formulate the ML solution w_{ML}.

Solution:

7.3.2 Maximum-A-Priori (MAP) Estimation

!! **Professor Q.:** *Linear regression: Prove and explain that L2-regularized linear regression (ridge regression) is equivalent to Maximum-A-Posteriori (MAP) estimation under the assumption of Gaussian output noise and with a Gaussian prior on the coefficients w*

- *Derive and formulate the MAP solution w_{MAP}*
- *assume that the output noise is Gaussian distributed – $1/\beta$ noise variance*
- *introduce a prior Gaussian distribution over the coefficients w – $1/\alpha$ variance*
- *express the posterior distribution over w,*
- *and calculate the MAP solution w_{MAP}*

Is the MAP w_{MAP} solution a fixed value or a distribution?

Guide: https://agustinus.kristia.de/techblog/2017/01/05/bayesian-regression/

Solution:

7.3.3 Bayesian Curve Fitting

Professor Q.: *Bayesian learning: Draw and explain a generative model (as a directed graphical model) in the supervised case, and give the predictive probability $p(t' \mid x', x, t)$ for a new output t', given a new input value x', and conditioned on the observed (training) data x, t.*

...

Professor Q.: *Draw and explain the Bayesian curve fitting (Bayesian linear regression) generative model as a directed graphical model.*

- *Focus on understanding the graphical model representation, for "learning" and "prediction".*
- *What are the key differences between Maximum Likelihood (ML) estimation and the Bayesian learning approach to estimate model weights vector?*

...

7.4 Summary

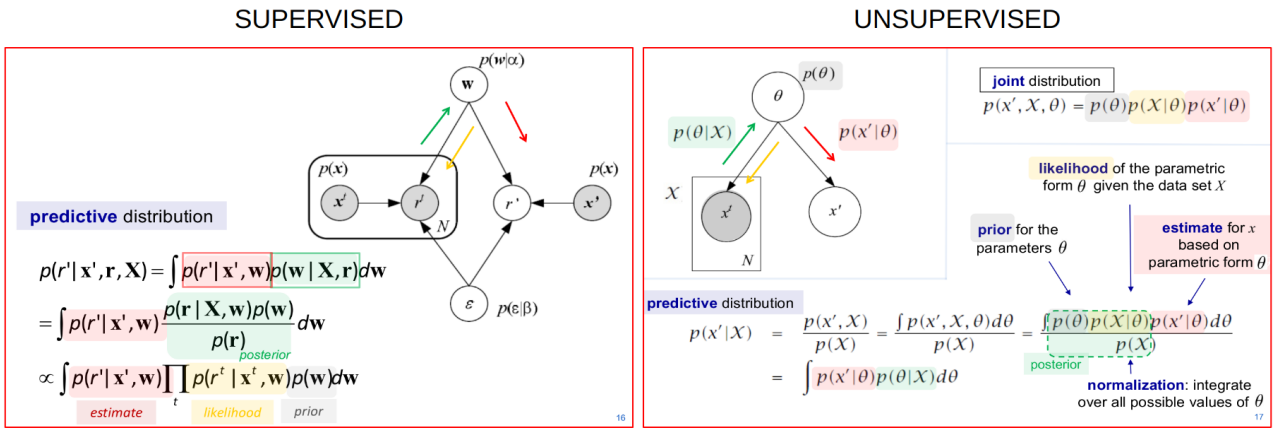


Figure 53: Images from lecture 7.

8 Hidden Markov Models and Gaussian Processes

"Markov chains are sequences of random variables in which the future variable is determined by the present variable but is independent of the way in which the present state arose from its predecessors." This work launched the theory of stochastic processes.

In the following, we will look at sequential problems from a Graphical Models perspective (remember, sequential data is not IID; however, it is possible to treat all observations as independent). The Markov assumption is that each observation only depends on the most recent previous observation (regardless of states in previous times). E.g., the simplest model

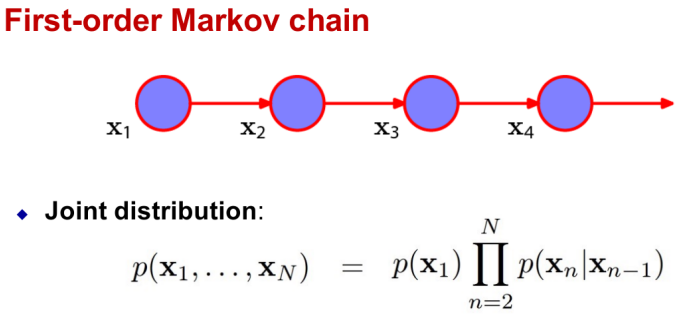


Figure 54: Images from lecture 8.

A second order would connect up to their second neighbor. When having N observations that can connect their first k neighbors, the parameters that the model must learn are $(k-1)k^N$; complexity time grows exponentially. It is possible to model the dependency of all previous observations with limited parameters by introducing a **state-space model**.

8.1 Hidden Markov Models

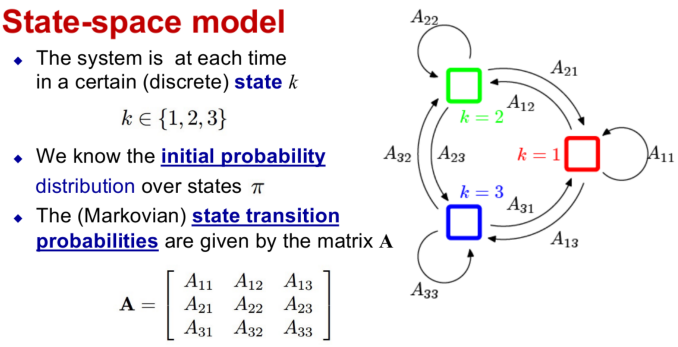


Figure 55: Image from lecture 8.

Notice that one cannot observe the states directly; they are hidden (latent). Furthermore, each state produces a characteristic output, given by an observation probability distribution over outputs φ.

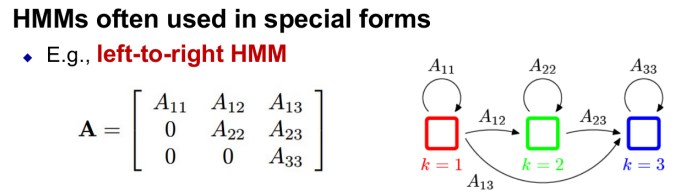


Figure 56: Image from lecture 8.

To be able to encode HMMs as graphical models, one needs to introduce latent variables z for the current system state so that the observed output x is conditioned on this state z.

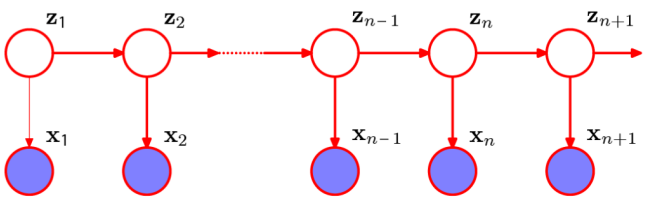


Figure 57: Image from lecture 8.

The (Markovian) state transition probabilities $p(z_n \mid z_{n-1})$ are given by the entries of the state transition matrix A; $p(z_{n,k} = 1 \mid z_{n-1,j} = 1) = A_{ij}$. With this, the number of parameters reduces to $nk(k-1) + const.$

General formulation of a Hidden Markov Model Recommend to watch lecture again

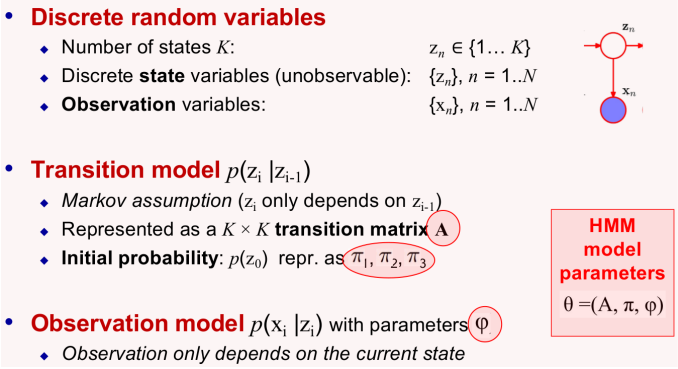


Figure 58: Image from lecture 8.

Hidden Markov Models prove particularly useful for sequential data and considering temporal correlations. Markov models work well if they have short and fixed-time dependencies. There are three possible operations that can be performed with HMMs:

- **Data likelihood** given a model and an observation (message passing scheme can be used).
- **Most likely state sequence**, just like the former, requires a model and an observation; furthermore, a message-passing scheme can be used.
- **Optimal HMM parameters**, Expectation Maximization is used for training the parameters.

Professor Q.: *Discuss the general formulation of a Hidden Markov Model (HMM) graphical model + max 15 lines. Wrap-up Slide.*

8.2 Introduction to Gaussian Processes

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

8.3 Summary

Keywords: state-space model,

9 Combining Multiple Learners – Ensembles

9.1 Introduction

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

9.2 Bagging, Boosting, and AdaBoost

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

9.3 Loss Function and Viola-Jones Face Detector

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

9.4 Decision Trees: CART Framework

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

9.5 Random Forests

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

9.6 Summary

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

10 Neural Networks and Feature Learning

10.1 Introduction

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

10.2 Multi-Layer Perceptrons (MLP)

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

10.2.1 Back Propagation

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

10.3 Convolutional Neural Networks (CNN)

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

10.4 Overfitting and Regularisation

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

10.5 Auto-encoders and Embeddings

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

10.6 Deep Learning Extensions

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

10.7 Recurrent Neural Networks

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

10.8 Summary

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

11 Ethical Aspects in Machine Learning

11.1 Examples of Powerful Algorithms and Ethical Concerns

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

11.2 Overconfidence and Unreliability of Models

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

11.3 Model Explainability

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor

semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

11.4 FAIRNESS: Criteria, Mitigation & "Fairness by Awareness"

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

11.5 Summary

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

References

[1] S. C. Society, *Simplifying the difference: Machine learning vs deep learning*, *https://www.scs.org.sg/articles/machine-learning-vs-deep-learning*, 2020.

[2] J. Fumo, "Types of machine learning algorithms you should know," *Medium*, Jun. 2017, [Accessed: Jul. 16, 2023]. [Online]. Available: *https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861*.

[3] O. Contreras Carrasco, "Gaussian mixture models explained - towards data science," *Medium*, Jun. 2019. [Online]. Available: *https://towardsdatascience.com/gaussian-mixture-models-explained-6986aaf5a95*.

[4] "Metacademy." [Accessed: Jul. 16, 2023], Metacademy.org. (2023), [Online]. Available: *https://metacademy.org/graphs/concepts/generative_vs_discriminative*.

[5] A. Lindholm, N. Wahlström, F. Lindsten, and T. B. Schön, *Machine Learning - A First Course for Engineers and Scientists*. Cambridge University Press, 2022. [Online]. Available: *https://smlbook.org*.

[6] S. Vignesh, "Parametric vs non parametric model selection for regression and classification based on statistical test," *Medium*, Aug. 2021, [Accessed: Jul. 16, 2023]. [Online]. Available: *https://22vignesh97.medium.com/parametric-vs-non-parametric-model-selection-for-regression-and-classification-based-on-statistical-152fddd4b384*.

[7] A. Rakhecha, "Importance of loss function in machine learning - towards data science," *Medium*, Sep. 2019, [Accessed: Jul. 16, 2023]. [Online]. Available: *https://towardsdatascience.com/importance-of-loss-function-in-machine-learning-eddaaec69519*.

[8] M. Vlastelica, "Learning theory: Empirical risk minimization - towards data science," *Medium*, Feb. 2019, [Accessed: Jul. 17, 2023]. [Online]. Available: *https://towardsdatascience.com/learning-theory-empirical-risk-minimization-d3573f90ff77*.

[9] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, Jul. 2014. DOI: *10.1017/CBO9781107298019*. [Online]. Available: *https://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/understanding-machine-learning-theory-algorithms.pdf*.

[10] P. Gomede Everton, "Independent and identically distributed (iid) in machine learning: Assumptions and implications," *Medium*, May 2023, [Accessed: Jul. 17, 2023]. [Online]. Available: *https://medium.com/@evertongomede/independent-and-identically-distributed-iid-in-machine-learning-assumptions-and-implications-930ee9821e14*.

[11] *Validation curve*, GeeksforGeeks, [Accessed: Jul. 18, 2023], Jul. 2020. [Online]. Available: *https://www.geeksforgeeks.org/validation-curve/*.

[12] J. Brownlee, *Step-by-step framework for imbalanced classification projects - machinelearningmastery.com*, Accessed May 07, 2023, 2020. [Online]. Available: *https://machinelearningmastery.com/framework-for-imbalanced-classification-projects/*.

[13] "Embeddings." [Accessed: Aug. 10, 2023], Google for Developers. (2022), [Online]. Available: *https://developers.google.com/machine-learning/crash-course/embeddings/video-lecture#:~:text=An%20embedding%20is%20a%20relatively,like%20sparse%20vectors%20representing%20words.*.

[14] T. Champion, M. Grześ, and H. Bowman, *Realising active inference in variational message passing: The outcome-blind certainty seeker*, 2021. arXiv: *2104.11798* [cs.LG].