Applied Machine Learning

Brooks Paige

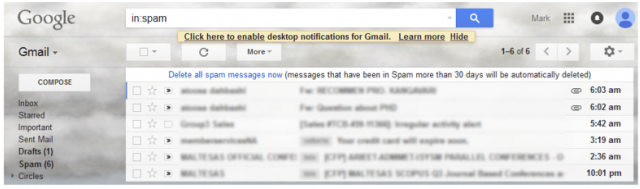
Week 1

What is applied machine learning?

- Not "theory" — asymptotic behavior, PAC bounds, ...
- Often:
  - how can we apply existing methods to new problems?
  - how can we scale existing (simple) methods to large data?
- How is ML used in industry? (When is ML not used?)
- What happens after a ML model is deployed in the "real world"?

What are machine learning applications?

- Spam filtering:



- Product recommendations:

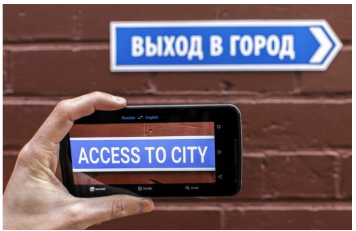
- Credit card fraud detection

Examples from https://www.cs.ubc.ca/~fwood/CS340/

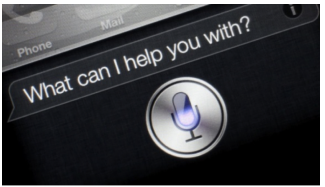
What are machine learning applications?

Motion capture:

Character recognition and machine translation:

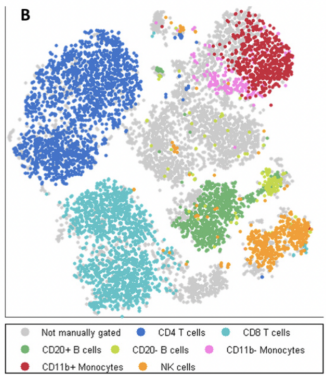


Voice recognition:

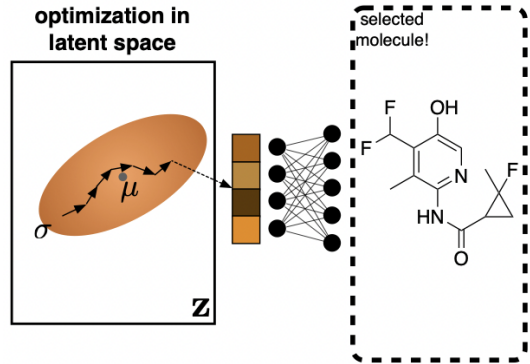


What are machine learning applications?

Identifying new cancer subtypes:



Developing new drugs:



Images: Amir et al. 2013; Bradshaw et al. 2019

Applied machine learning, bird's eye view

1. Collect data.
2. Fancy machine learning!
3. Profit?

Applied machine learning in practice

1. Learn about the domain / application
2. Identify an appropriate task
3. **Collect data**
4. Clean and preprocess data
5. Transform data or select useful features
6. Choose an appropriate method
7. **Fancy machine learning!**
8. Evaluate, visualize, and interpret results
9. **Profit?**

Applied machine learning in practice

1. Learn about the domain / application
2. Identify an appropriate task
3. **Collect data**
4. Clean and preprocess data
5. Transform data or select useful features
6. Choose an appropriate method
7. **Fancy machine learning!**
8. Evaluate, visualize, and interpret results
9. **Profit?**

Also, typically you'll go through **repeated cycles of these steps** ...

Applied machine learning in practice

1. **Learn about the domain / application**
2. **Identify an appropriate task**
3. **Collect data**
4. Clean and preprocess data
5. Transform data or select useful features
6. Choose an appropriate method
7. **Fancy machine learning!**
8. Evaluate, visualize, and interpret results
9. **Profit?**

Applied machine learning in practice

1. Learn about the domain / application
2. Identify an appropriate task
3. **Collect data**
4. **Clean and preprocess data**
5. **Transform data or select useful features**
6. Choose an appropriate method
7. **Fancy machine learning!**
8. Evaluate, visualize, and interpret results
9. **Profit?**

Applied machine learning in practice

1. Learn about the domain / application
2. Identify an appropriate task
3. **Collect data**
4. Clean and preprocess data
5. Transform data or select useful features
6. **Choose an appropriate method**
7. **Fancy machine learning!**
8. Evaluate, visualize, and interpret results
9. **Profit?**

Applied machine learning in practice

1. Learn about the domain / application
2. Identify an appropriate task
3. **Collect data**
4. Clean and preprocess data
5. Transform data or select useful features
6. Choose an appropriate method
7. **Fancy machine learning!**
8. **Evaluate, visualize, and interpret results**
9. **Profit?**

Applied machine learning in practice

1. Learn about the domain / application
2. Identify an appropriate task
3. **Collect data**
4. Clean and preprocess data
5. Transform data or select useful features
6. Choose an appropriate method
7. **Fancy machine learning!**
8. Evaluate, visualize, and interpret results
9. **Profit?** (. . . and **monitor performance after deployment**)

Learn about the domain and identify a task

. . . Lots of work to do before collecting any data!

- For all of those example applications, the first step was to **identify a problem** that can be **solved with data**.

Learn about the domain and identify a task

. . . Lots of work to do before collecting any data!

- For all of those example applications, the first step was to **identify a problem** that can be **solved with data**.
- **This is hard.** (MSc students will experience this challenge a bit in narrowing down a thesis project!)

Learn about the domain and identify a task

. . . Lots of work to do before collecting any data!

- For all of those example applications, the first step was to **identify a problem** that can be **solved with data**.
- **This is hard.** (MSc students will experience this challenge a bit in narrowing down a thesis project!)
- In this module, we typically assume that the data already exists, in some form, and that we have a clearly defined task.
  - In real world or industry settings, this might actually be a lot of the work!

What is data?

What is data?

Most data comes in a **data frame** like the one on the right.

- Columns are per-instance attributes:
  - e.g. age, height, eye color, column ID, ...
  - Also called: feature, variable, field, ...
- Rows are instances
  - Also called: object, data point, sample, ...

Several steps before $\mathbf{X} \in \mathbb{R}^{N \times P}$...!

Attributes

Objects

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Table from Tan, Steinbach, Kumar, "Introduction to Data Mining"

Data types

- Numerical (real-valued)
- Categorical
- Ordinal
- Dates and times
- Coordinates (geospatial...)
- Text
- Unique identifiers
- Images, video, audio, ...

Numerical feature pre-processing

Pre-processing

- Scaling and centering (whitening)
- Rank tranformation
- Clipping
- Non-linear transformations (e.g. $\log$ or $\sqrt{\ldots}$)

Numerical feature pre-processing

Pre-processing

- Scaling and centering (whitening)
- Rank tranformation
- Clipping
- Non-linear transformations (e.g. $\log$ or $\sqrt{\ldots}$)

To consider: What machine learning models are sensitive to scaling? (What ones are not?)

Numerical feature pre-processing

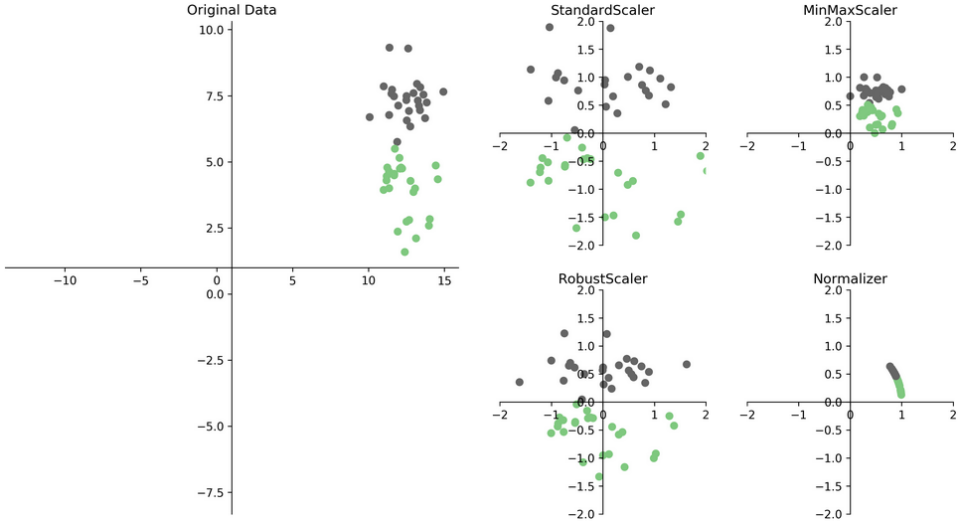


Figure: Andreas Müeller

Numerical feature pre-processing

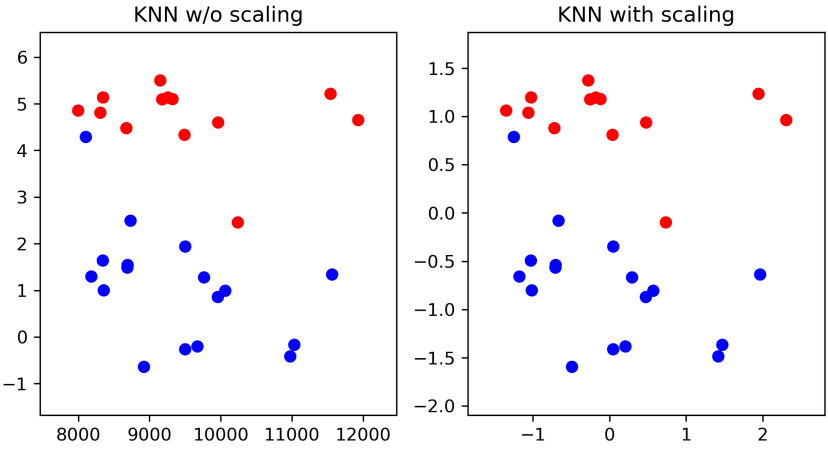


Figure: Andreas Müeller

Numerical feature pre-processing

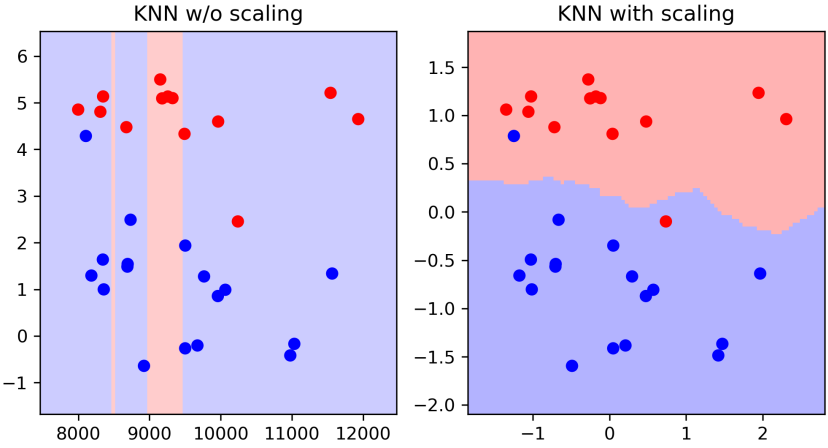


Figure: Andreas Müeller

Numerical feature pre-processing

Other thoughts:

- Be careful if scaling or centering **sparse** data!
- Fixed transforms are useful for addressing assumptions about domains, e.g.

$$\log(x) : \mathbb{R}^+ \to \mathbb{R}$$

$$\text{logit}(x) : [0, 1] \to \mathbb{R} \qquad \left(\text{logit}(x) \equiv \log \frac{x}{1 - x} \right)$$

- Sometimes helps with normality assumptions as well ...

Categorical feature pre-processing

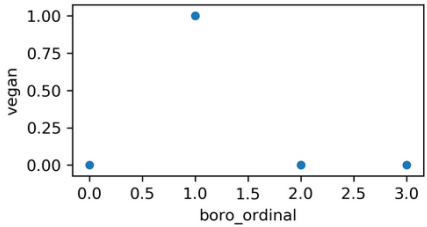
- One-hot encoding
- Label encoding
- Frequency encoding
- Target encoding
- Learn embeddings

	boro	salary	vegan
0	Manhattan	103	No
1	Queens	89	No
2	Manhattan	142	No
3	Brooklyn	54	Yes
4	Brooklyn	63	Yes
5	Bronx	219	No

Example: Andreas Müller

Categorical feature pre-processing

	boro	salary	vegan
0	2	103	No
1	3	89	No
2	2	142	No
3	1	54	Yes
4	1	63	Yes
5	0	219	No



Label or **ordinal** encoding — makes sense if there really is a natural order...

Categorical feature pre-processing

	boro	salary	vegan
0	Manhattan	103	No
1	Queens	89	No
2	Manhattan	142	No
3	Brooklyn	54	Yes
4	Brooklyn	63	Yes
5	Bronx	219	No

	salary	vegan	boro_Bronx	boro_Brooklyn	boro_Manhattan	boro_Queens
0	103	No	0	0	1	0
1	89	No	0	0	0	1
2	142	No	0	0	1	0
3	54	Yes	0	1	0	0
4	63	Yes	0	1	0	0
5	219	No	1	0	0	0

One-hot or **dummy variable** encoding

Categorical feature pre-processing

Other ideas:

- **mean** encodings replace the categorical variable with some other statistic, e.g. the mean of the target variable
  - ...many potential pitfalls
- **Frequency** (or count) encodings record how often a particular value occurs
  - ...e.g. "Bag-of-words" representations for text
  - We'll come back to this later in the module
- Learned encodings / **embedding layers** in neural networks map each discrete value onto a vector in $\mathbb{R}^D$
  - ...a good option if trained end-to-end with the rest of the model
  - common in NLP (e.g. word or character embeddings)

How do we learn from data?

The combination of some data and an aching desire for an answer does not ensure that a reasonable answer can be extracted from a given body of data.

– John Tukey

Central goal of machine learning

In machine learning, we care about the error on test data!

Central goal of machine learning

In machine learning, we care about the error on test data!

This is different than the error on our **training** data.
Analogy to sitting an exam:

- Training error is the score on a practice exam
- Test error is the score on the real exam
- We want to do well on the **real** exam, not the practice one.

Memorizing the practice exam will produce a low training error.
Learning is necessary to also have a low test error!

Fitting a polynomial

8 data points,
M parameters:

$$f(x) = \sum_{i=1}^{M} \beta_i x^i$$

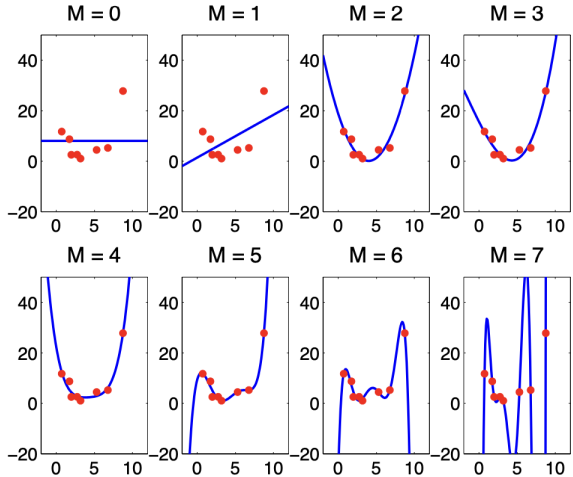


Figure: Zoubin Ghahramani

Why does machine learning work?

Fundamental assumption: **Test data is similar to training data**

- If the training data and test data are unrelated, then we can't learn anything
- Generalization requires assumptions!

Training error and generalization

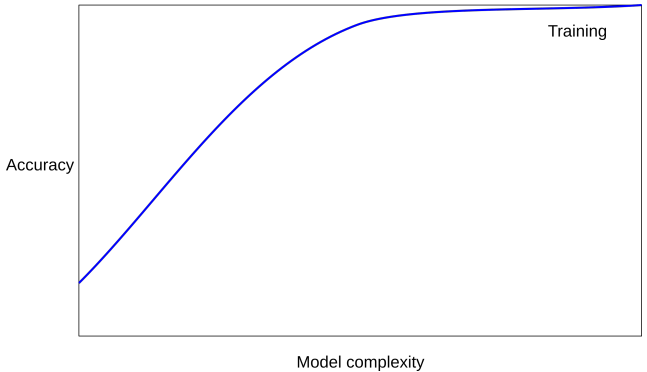


Figure: Andreas Müller

Training error and generalization

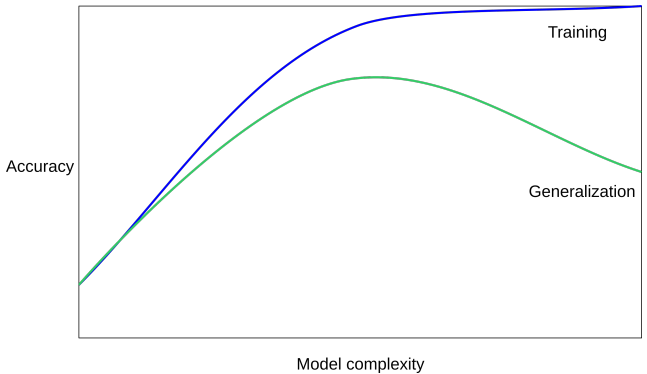


Figure: Andreas Müller

Training error and generalization

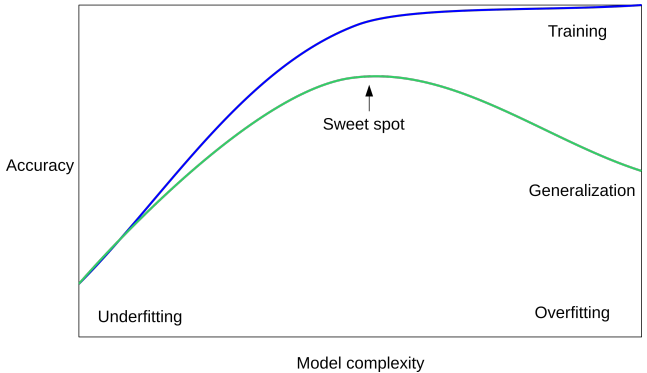


Figure: Andreas Müller

Don't overfit to the test set!

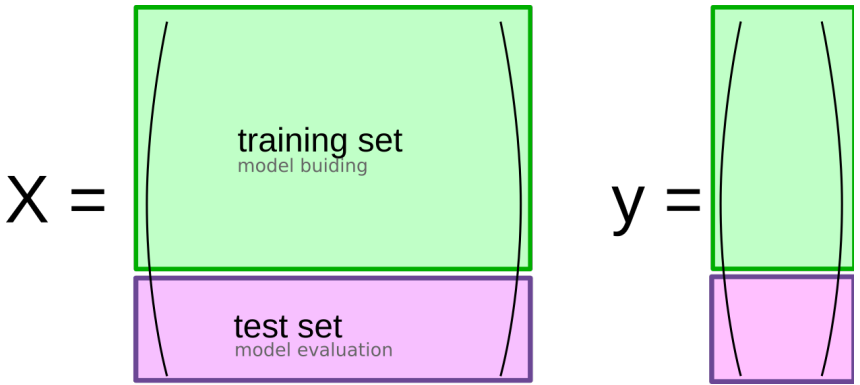


Figure: Andreas Müller

Don't overfit to the test set!

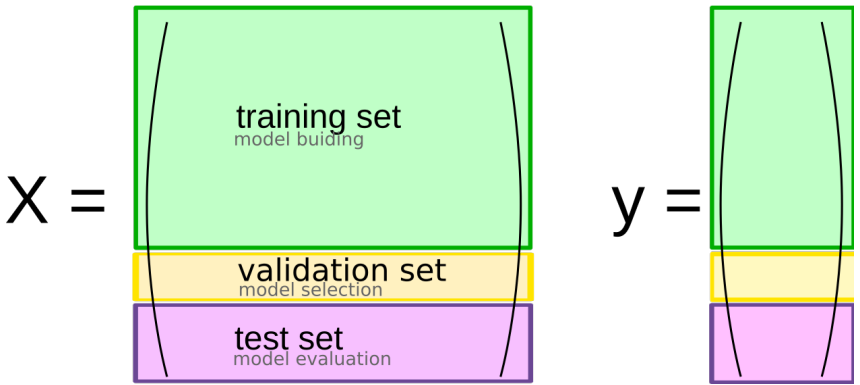


Figure: Andreas Müller

Cross-validation

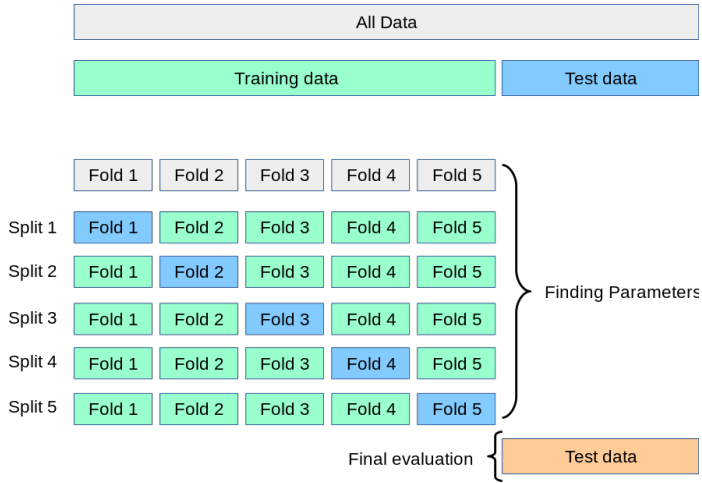


Figure: scikit-learn

Cross-validation

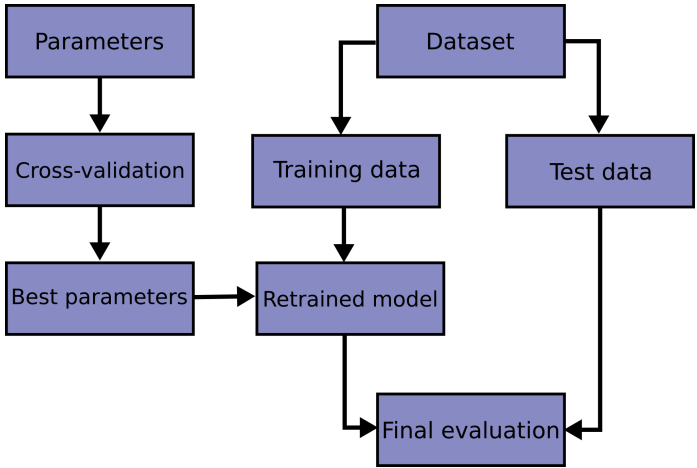


Figure: scikit-learn

Validation sets for non-iid data

Data is often not independent and identically distributed (*iid*)! A **random split** might not be appropriate.

- **Grouped data**: e.g. data collected from different cities, classrooms, etc, where we expect to see higher correlation within-group than between groups
  - **Stratified sampling** of cross-validation sets
- **Time-series data**: in a random split, you are effectively testing interpolation between points, instead of extrapolation into the future
  - Temporal splits ensure validation / testing only on held-out future data
  - **Careful:** more data than you think is actually time series data! Data is typically collected sequentially over a long period of time (even if you don't know it)

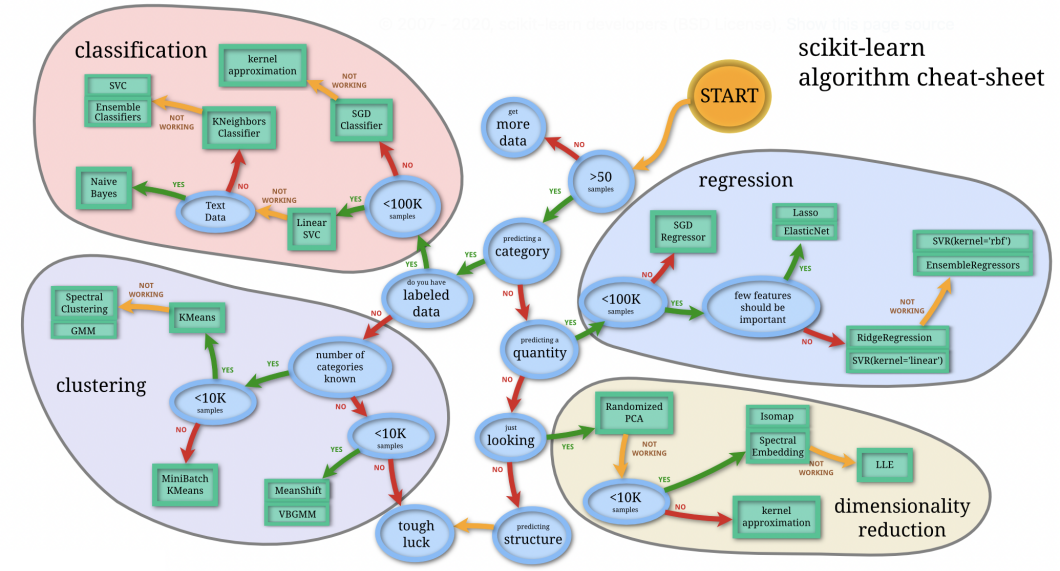
Pitfalls

Don't look at the test set!

- It's easy to accidentally look at the test set! Don't do it — you won't be able to trust your results
- Example: computing statistics for scaling transforms
- Example: visualizations and exploratory data analysis

Choosing a model

Machine learning as a bag of tricks



How much data do I need?

Two types of data sources

Cheap (infinite) data

Predict observable events

- Ad clicks
- Product recommendations
- Stock market
- House numbers
- Transcription autocorrect
- ...

Expensive data

Automate complex processes

- Medical diagnosis
- Drug trial
- Microchip design
- ...

Big data headaches

More data is not always better

- Is dealing with the large size worth the hassle? Data scientist / analyst time is expensive!
  - (can you get away with just using a subset that will fit in RAM?)

Big data headaches

More data is not always better

- Is dealing with the large size worth the hassle? Data scientist / analyst time is expensive!
  - (can you get away with just using a subset that will fit in RAM?)
- Where did the data come from? Is it representative of future data?

Big data headaches

More data is not always better

- Is dealing with the large size worth the hassle? Data scientist / analyst time is expensive!
  - (can you get away with just using a subset that will fit in RAM?)
- Where did the data come from? Is it representative of future data?
- Would a smaller dataset of higher-quality data be more useful, or less useful?
  - (this can be situation dependent!)

Evaluation and metrics

Evaluation and metrics

(“did it work?”)

Metrics for classification

Accuracy: what fraction of examples are classified correctly?

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{I}[f(x_i) = y_i]$$

Metrics for classification

Accuracy: what fraction of examples are classified correctly?

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{I}[f(x_i) = y_i]$$

- Hard predictions! Does not take into account uncertainty

Metrics for classification

Accuracy: what fraction of examples are classified correctly?

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{I}[f(x_i) = y_i]$$

- Hard predictions! Does not take into account uncertainty
- Can't be optimized with gradient-based methods

Metrics for classification

Accuracy: what fraction of examples are classified correctly?

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{I}[f(x_i) = y_i]$$

- Hard predictions! Does not take into account uncertainty
- Can't be optimized with gradient-based methods
- Misleading for imbalanced classes

Metrics for classification

How do we choose a threshold value for "soft" classifiers?

- Suppose $f(x_i)$ returns some real-value (as opposed to a label $\hat{y}_i \in \{0, 1\}$).

Metrics for classification

How do we choose a threshold value for "soft" classifiers?

- Suppose $f(x_i)$ returns some real-value (as opposed to a label $\hat{y}_i \in \{0, 1\}$).
- We can define a classifier as

$$\hat{y}_i = \begin{cases} 1 & f(x_i) \geq c \\ 0 & f(x_i) < c \end{cases}$$

where c is a threshold or cutoff parameter.

Metrics for classification

How do we choose a threshold value for "soft" classifiers?

- Suppose $f(x_i)$ returns some real-value (as opposed to a label $\hat{y}_i \in \{0, 1\}$).
- We can define a classifier as

$$\hat{y}_i = \begin{cases} 1 & f(x_i) \geq c \\ 0 & f(x_i) < c \end{cases}$$

  where c is a threshold or cutoff parameter.
- Different values lead to different trade-offs between **false positives** and **false negatives**!

Understanding classifier performance

Two classes: positive and negative.

- **Precision**: "How many of those predicted positive are actually positive?"

$$\frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

- **Recall** or **sensitivity**: "How many of those which are actually positive are correctly predicted as positive?"

$$\frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

- **Specificity**: "How many of those which are actually negative are correctly predicted as negative?"

$$\frac{\text{true negatives}}{\text{true negatives} + \text{false positives}}$$

Considering classifier trade-offs

Solution: look at the area under the curve (**AUC**)

- True positive vs false positive
- Precision vs recall
- ...

Exercise: think about how this is affected by class imbalance

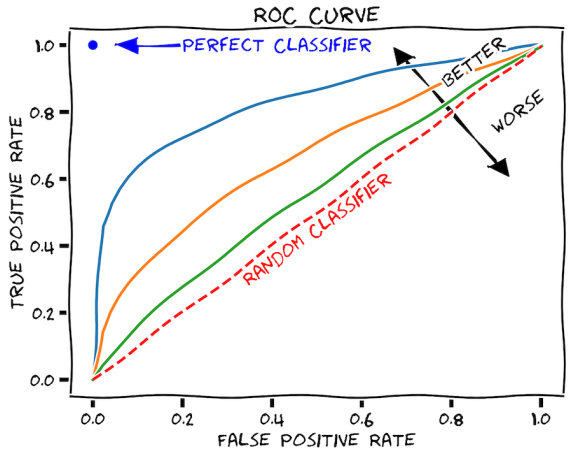


Figure: Martin Thoma

Classifier performance: log-loss

Maximum likelihood estimation:

- Define $f_\theta(x_i)$ which returns a value in the range $[0, 1]$
- Interpret this output as a probability $p(y_i = 1|x_i; \theta)$
- Optimize the parameters θ by maximizing the (log-)probability of the labels given the data:

$$\begin{aligned}\hat{\theta} &= \arg\max_\theta \sum_{i=1}^{N} \log p(y_i = 1|x_i; \theta) \\ &= \arg\max_\theta \sum_{i=1}^{N} y_i \log f_\theta(x_i) + (1 - y_i) \log(1 - f_\theta(x_i))\end{aligned}$$

This (with flipped sign) is also called **binary cross-entropy** loss.

Metrics for regression

Mean squared error (MSE):

$$\frac{1}{N}\sum_{i=1}^{N}(f(x_i) - y_i)^2$$

- Easy to optimize directly! (gradients, second-order methods)
- RMSE $= \sqrt{\text{MSE}}$
- Analogous to assuming normal-distributed errors
- Exercise: what is the best constant predictor?

Metrics for regression

Mean absolute error (MAE):

$$\frac{1}{N}\sum_{i=1}^{N}|f(x_i)-y_i|$$

- Also easy to optimize directly (though not by second-order methods)
- Less sensitive to outliers
- Analogous to assuming Laplace-distributed errors
- Exercise: what is the best constant predictor?

Also this week:

Mathematics refresher

Mathematics refresher

David Barber, *Bayesian Reasoning and Machine Learning* (BRML)

Online at `http://www.cs.ucl.ac.uk/staff/d.barber/brml/`

- Chapter 1 (Probabilistic reasoning)
- Appendix A (Background mathematics)
- Slides (from David) on Moodle

How (not) to do data science

1. there's no need to look at the data before running a model (the AI will handle it)
2. pick a model based on hype or social media mentions
3. don't worry about downstream tasks, users of the model, or what happens after deployment — just maximize "accuracy"
4. don't bother comparing performance against simple baselines