BROWN
Computer Science

CS1951A: Data Science

Lecture 14: Introduction to Machine Learning

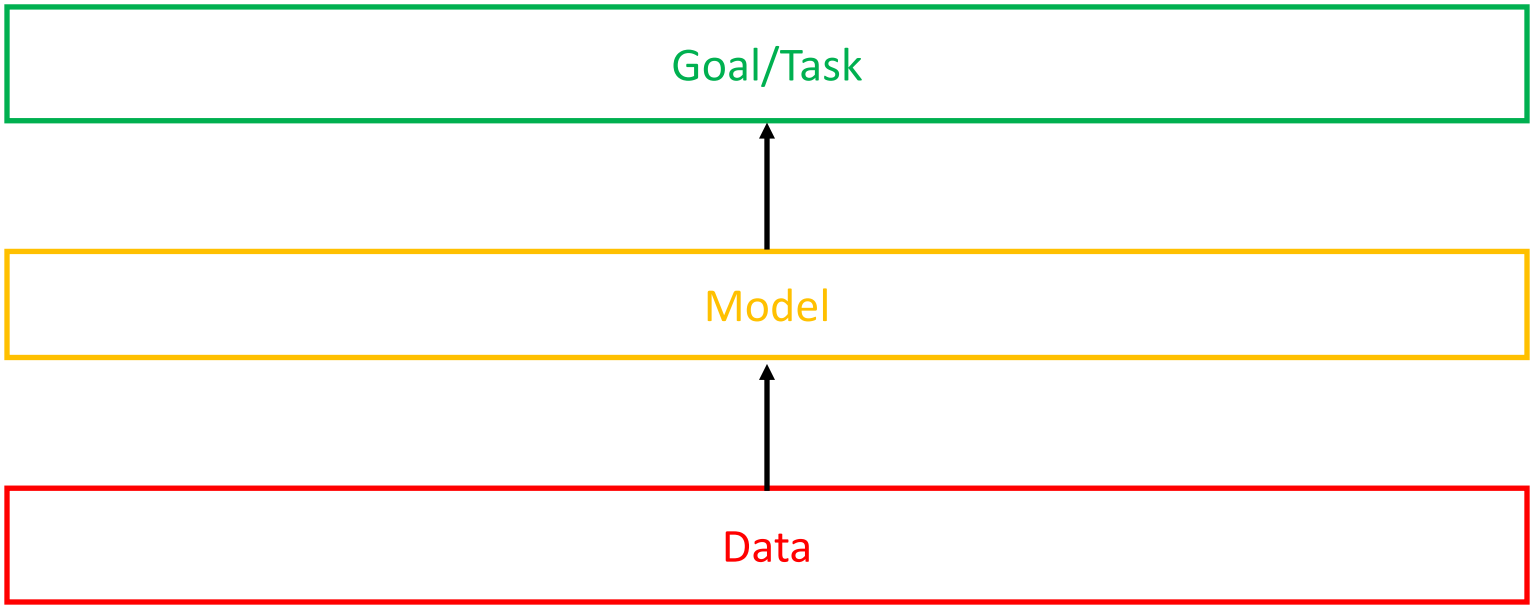
Lorenzo De Stefani

Spring 2022

Outline

- ML "preliminaries"—terminology, basic building blocks, conceptual background
- Choosing the candidate models/predictors
- Evaluating the performance of a ML algorithm
- Supervised vs unsupervised learning
- Clustering with K-means

Simplified ML



Simplified ML

Task: Prediction of some kind, e.g.:

- price of a stock (number)
- criterion to decide membership in a class (classification)
- sentiment of a piece of text (discrete label)
- objects in an image (tagging)
- strategy for a video game (sequence)
- parse tree of a sentence (tree structure)

Model

Data

Simplified ML

Task: Prediction of some kind

Model

Data: records selected from underlying population

- Can be anything. Usually, data size of available data and/or representation is the limiting factor.
- Generally, we assume that the data available is a randomly selected "sample" of a larger population
- Data point assumed to be selected independently and uniformly at random
- Often called the learning or training sample

Simplified ML

Task: Prediction of some kind

Model: How is the problem is structured? How to estimate parameters?

- Start asking the question: what do I believe the answer is? What Is the right model for the task at hand?
  - We select the set of candidate models/predictors to be evaluated
    - Linear/Logistic regression
    - SVMs
    - Naive Bayes, Bayesian Networks
    - Neural Networks
  - We introduce "selection bias". We can only select the "best" out of the option we decide to consider.
- How complex should my model be?
  - More complex -> more expressive but "harder to learn"

Data

Simplified ML

Task: Prediction of some kind

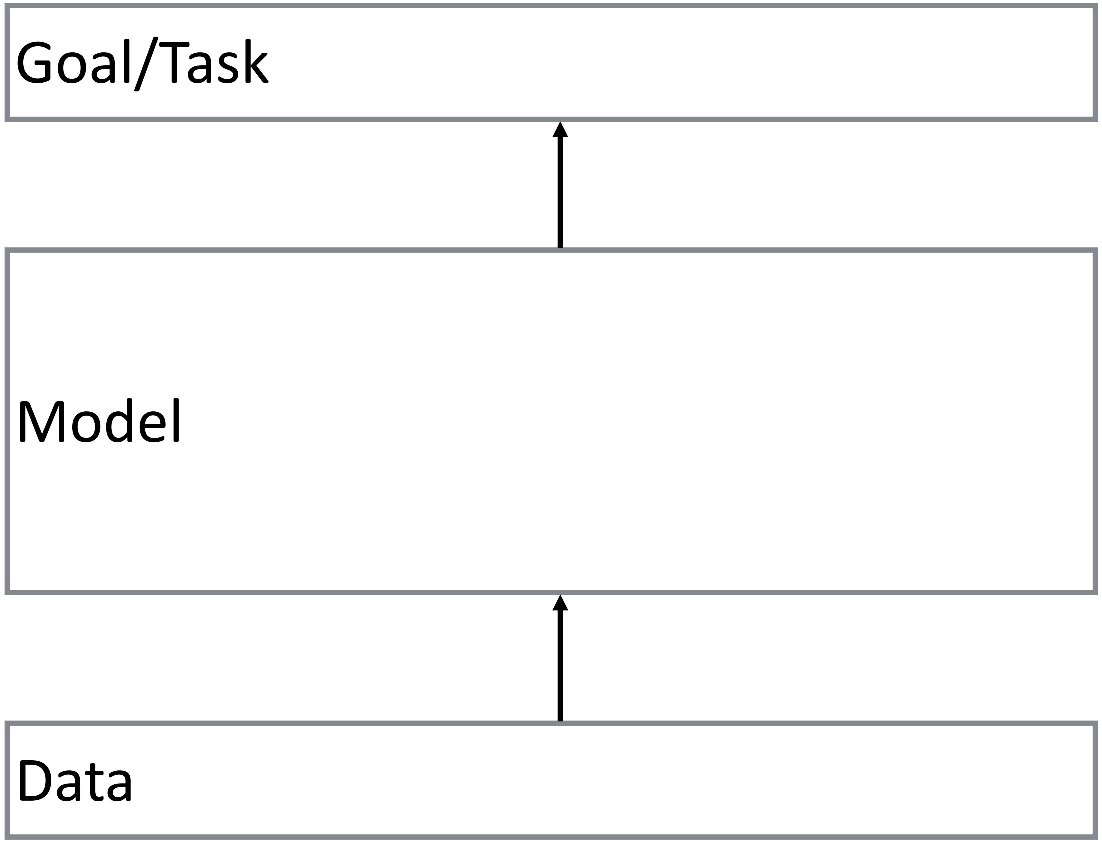
Model: How is the problem is structured? How to estimate parameters?

Data

GOAL: Use the data to select a model/function that can be used for our task and that generalized to the larger underlying population

- We want to learn insights that still hold true for the overall population, not just the observed part
- Avoid overfitting to the sample

Defining an ML problem



Defining an ML problem

Task = Increase consumption of advertisement of a website

Model

Data

Defining an ML problem

Task = Increase consumption of advertisement of a website

↑

Model

↑

Data: browsing information collected from willing ☺ users

Defining an ML problem

Task = Increase consumption of advertisement of a website

Formal definition in terms of:

- objective function: a quantity I want to maximize minimize to represent quality if selected model
- loss function: difference between selected model and best possible model among the available ones. This should be minimized!

Model

Data: browsing/clicking history

Modeled as vectors whose features represent information
E.g., attributes of user, clicks per ads, has photos....

Prediction Target

- Goal = Increase consumption of advertisment for your website
- Objective function….ideas?
  - Time spent on site (avg. per user/total)
  - Number of users
  - Number of pages read (need to define “read”)
  - Number of ads clicked on ✓
  - Time per page
  - Pages shared…

Loss function

- Let us say the goal of my ML task is to design a predictor that for a given ad gives me a guess of the number of clicks that it will receive

- The loss function will be the error of the prediction with respect to the true value

  – Absolute/quadratic differece

Features

- Data = Information on ads comsumption collected using user browsing data

- Think of it as "examples" to learn from. Generally called training data

- The information expressed in them is what our learning algorithms will base its decision on which is a good predictor

- Examples:
  - Article topic
  - Recency (minutes since release)
  - Words in title/snippet
  - Presence of photo
  - Reading level
  - Fonts/layouts
  - User location
  - Topics of articles the user has read previously
  - Number of likes

Features

Clicks	Recency	Reading Level	Photo	Title
10	1.3	11	1	“New Tax Guidelines”
1000	1.7	3	1	“This 600lb baby…”
1000000	2.4	2	1	“18 reasons you should *never* look at this cat unless you…”
1	5.9	19	0	“The Brothers Karamazov: a neo-post-globalist perspective”

Features

Clicks	Recency	Reading Level	Photo	Title
10	1.3	11	1	"New Tax Guidelines"
1000	1.7	3	1	"This 600lb baby…"
1000000	2.4	2	1	"18 reasons you should *never* look at this cat unless you…"
1	5.9	19	0	"The Brothers Karamazov: a neo-post-globalist perspective"

Y: What we are constructing a predictor for

- Think of it as our independent variable

Features

Clicks	Recency	Reading Level	Photo	Title
10	1.3	11	1	"New Tax Guidelines"
1000	1.7	3	1	"This 600lb baby…"
1000000	2.4	2	1	"18 reasons you should *never* look at this cat unless you…"
1	5.9	19	0	"The Brothers Karamazov: a neo-post-globalist perspective"

X: the input of the predictor f we are building

- Our independent variables

$$Y = f(X)$$

Features

numeric features — defined for (nearly) every row

Clicks	Recency	Reading Level	Photo	Title
10	1.3	11	1	"New Tax Guidelines"
1000	1.7	3	1	"This 600lb baby…"
1000000	2.4	2	1	"18 reasons you should *never* look at this cat unless you…"
1	5.9	19	0	"The Brothers Karamazov: a neo-post-globalist perspective"

Features

boolean features — 0 or 1 (dummy variables)

Clicks	Recency	Reading Level	Photo	Title
10	1.3	11	1	"New Tax Guidelines"
1000	1.7	3	1	"This 600lb baby…"
1000000	2.4	2	1	"18 reasons you should *never* look at this cat unless you…"
1	5.9	19	0	"The Brothers Karamazov: a neo-post-globalist perspective"

Features

Strings and text

Clicks	Recency	Reading Level	Photo	Title
10	1.3	11	1	“New Tax Guidelines”
1000	1.7	3	1	“This 600lb baby…”
1000000	2.4	2	1	“18 reasons you should *never* look at this cat unless you…”
1	5.9	19	0	“The Brothers Karamazov: a neo-post-globalist perspective”

Features

strings and boolean features — 0 or 1 (“dummy” variables)

Clicks	Recency	Reading Level	Photo	Title: “new”	Title: “tax”	Title: “this”	Title: “…”	…
10	1.3	11	1	1	0	0	0	…
1000	1.7	3	1	0	0	1	1	…
1000000	2.4	2	1	0	0	1	1	…
1	5.9	19	0	0	0	0	0	…

Features

sparse features — 0 for most entries

Clicks	Recency	Reading Level	Photo	Title: "new"	Title: "tax"	Title: "this"	Title: "…"	…
10	1.3	11	1	1	0	0	0	…
1000	1.7	3	1	0	0	1	1	…
1000000	2.4	2	1	0	0	1	1	…
1	5.9	19	0	0	0	0	0	…

Defining an ML problem

Objective/Loss Function = squared difference between predicted total number of clicks and actual total number of clicks

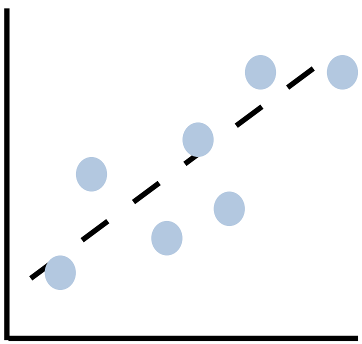
↑

Model

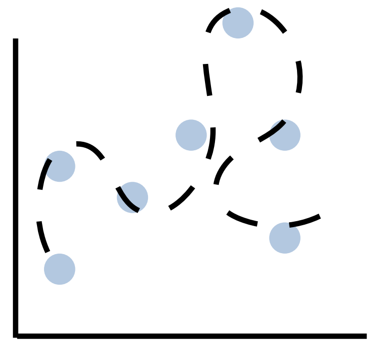
↑

Features = {Recency:float, ReadingLevel:Int, Photo:Bool, Title_New:Bool, Title_Tax:Bool, …}

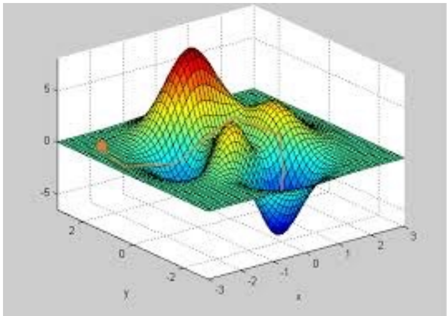
Family of predictors



Linear predictor



Polynomial predictor



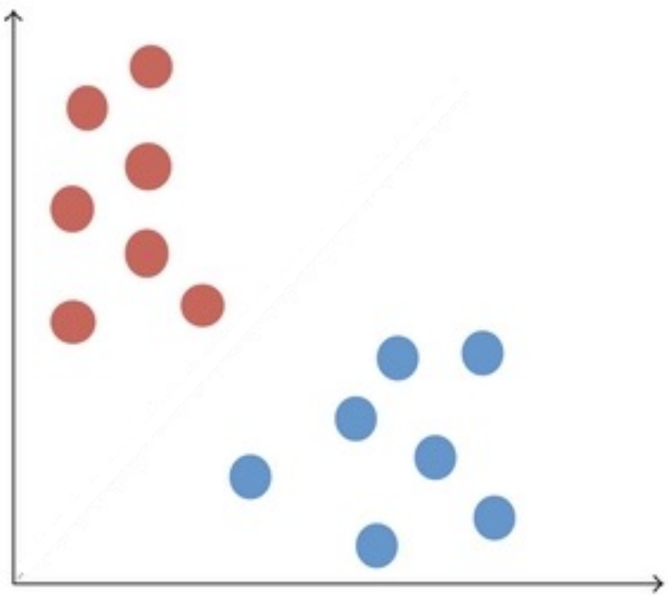
Arbitrarily complex predictor

We need to set the possible functions/predictors among which we will choose a best predictor

- Referred as ”Family of candidate predictors/functions”
- Can be finite or infinite
  - E.g., half-planes, polygons, non-linear modes,...
- Our choice of the candidate family of functions introduces selection bias
  - We can only select a predictor among those we are looking for!

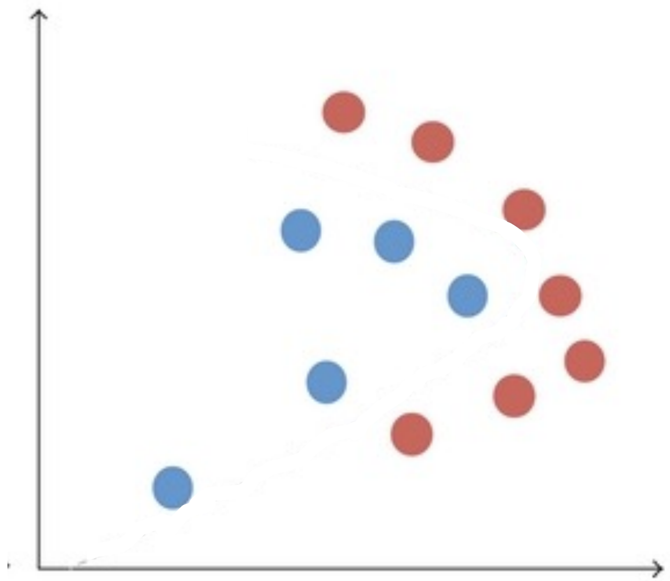
The complexity vs generalizability tradeoff

- How "complex" should the candidate predictors we evaluate be?



The complexity vs generalizability tradeoff

- How "complex" should the candidate predictors we evaluate be?



The complexity vs generalizability tradeoff

- Intuition: the complexity of a model, or a class or models, is tied to how expressive it can be
  - How "complicated/complex" is the function they compute/realize
  - How many "special cases" it can accommodate
- Many different notions of complexity in the ML literature:
  - Cardinality, degree of polynomial, VC dimension, Rademacher complexity,.....

The complexity vs generalizability tradeoff

- The more complex the model, the more expressive
  - Captures more details about the model
- The more complex the model, the harder it is to "learn it"
  - The more examples we need to see
  - The more information we need to acquire
- While using complex models may seem appealing, we incur in the risk of overfitting to the data
  - We need to observe a high number of examples to have the same guarantees as if we had simpler models

ML: From a practical point of view

- Input data/features need to be concrete and representable.
- Definition of "success" needs to be quantifiable
  - Generally, should be expressed as a differentiable mathematical function
- Learning algorithm should be feasible
  - Run in polynomial time with respect to the number of models being considered, the amount of data points in the training set and/or some parameters od the task

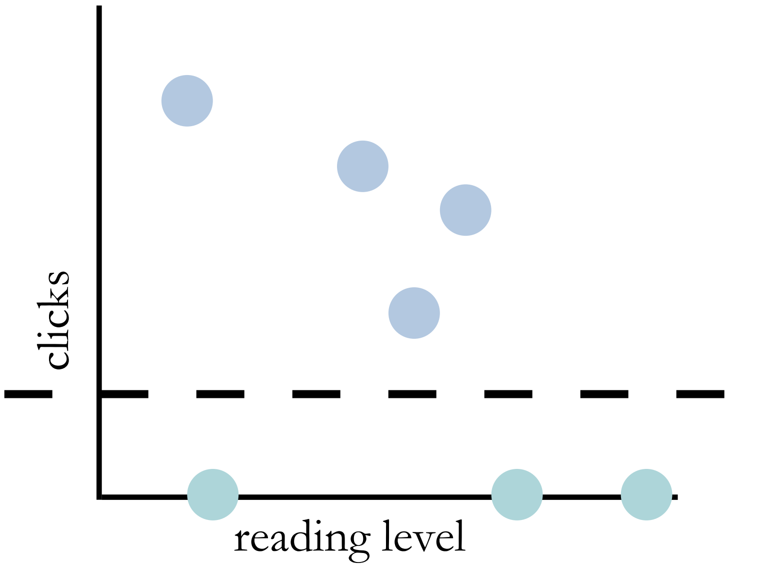
ML: From a probabilistic point of view

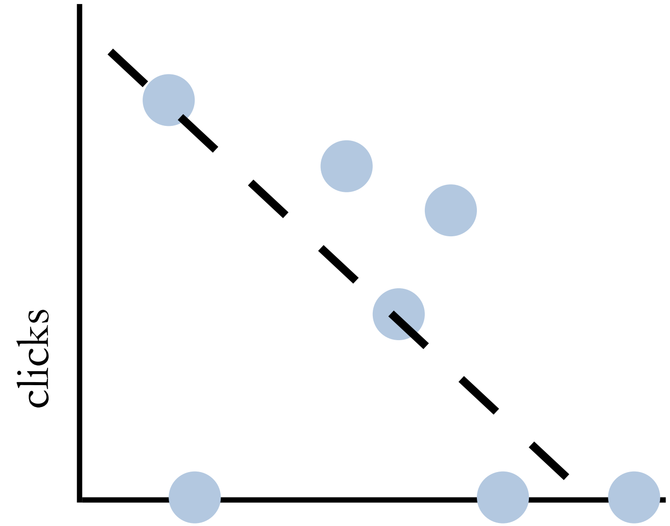
- We need to have reasonable assumptions on the data acquisition mechanism
  - Independent/identically distributed samples
- Success tied to a notion of "generalizability"
  - Insight obtained on the sample should also hold – generalize- for the entire population
  - E.g, suppose we want to select a predictor f^* for a value μ of interest using the training data
  - In Statistical Learning (PAC learning) we desire guarantees of the type

$$P(|\mu - f^*| > \epsilon) \leq \delta$$

    - ϵ accuracy: how much error we are tolerating
    - $\delta \in (0,1)$ confidence: the probability of our prediction being correct

Classification and Regression





The predictor partitions the points in classes

- Assigns a "label" associate with the class
  - Discrete output
- Binary classification with two classes
  - E.g., "clicked, not clicked"

$f(\mathit{reading\ level}) = \{\mathit{clicked}, \mathit{not\ clicked}\}$

- Multi-class classification

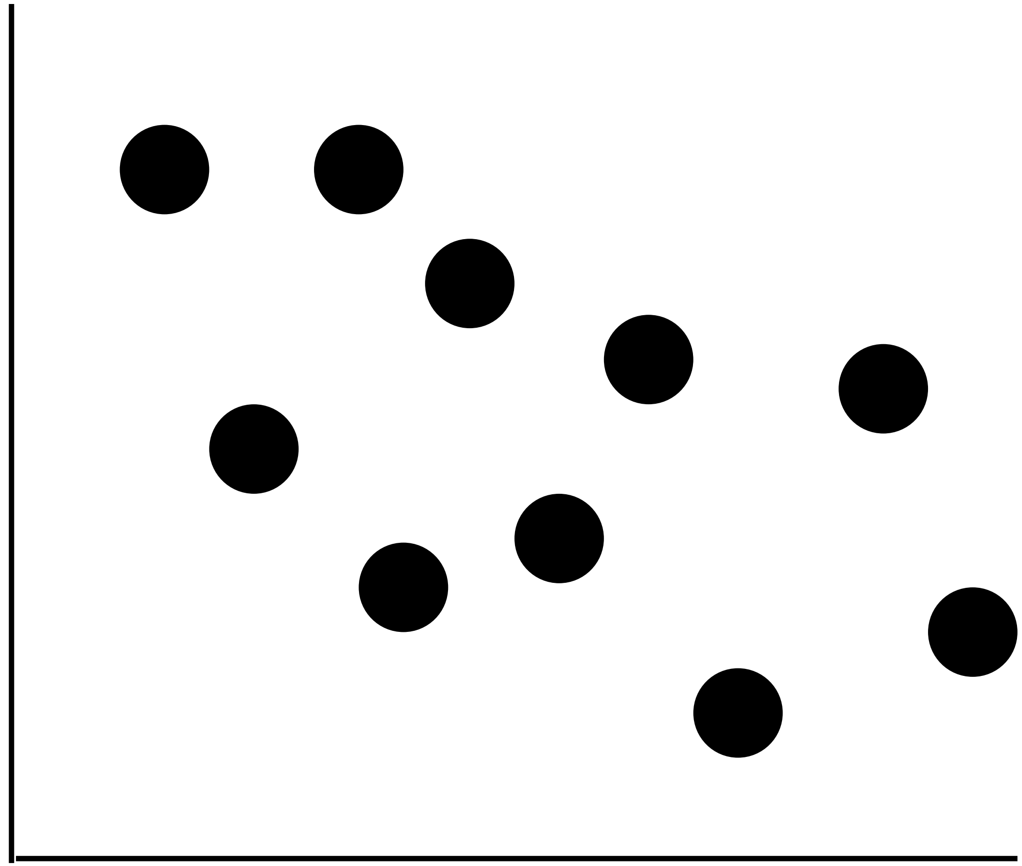
The predictor provides an actual estimate of the value of interest

- Returns real values
- $\mathit{clicks} = m(\mathit{reading\_level}) + b$
- m and b are the parameters of the model to be estimated

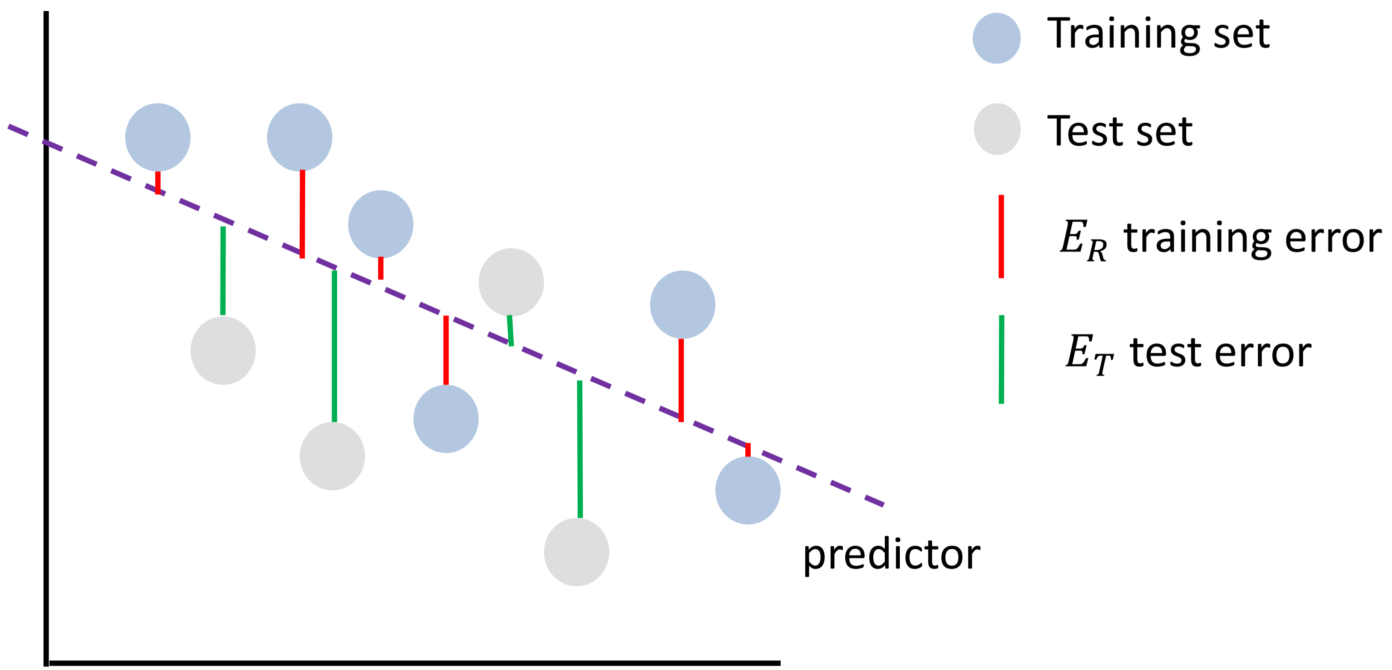
Evaluating the performance of a predictor

- Once I select a classifier/predictor from my class I may want to evaluate or validate its performance
- This operation should always be performed using data distinct from that used to select (train) the model itself!
  - “Train” using training data
  - “Validate” using test/validation data
  - Both sets are assumed to be obtained from the same random process/population

Train/Test Splits

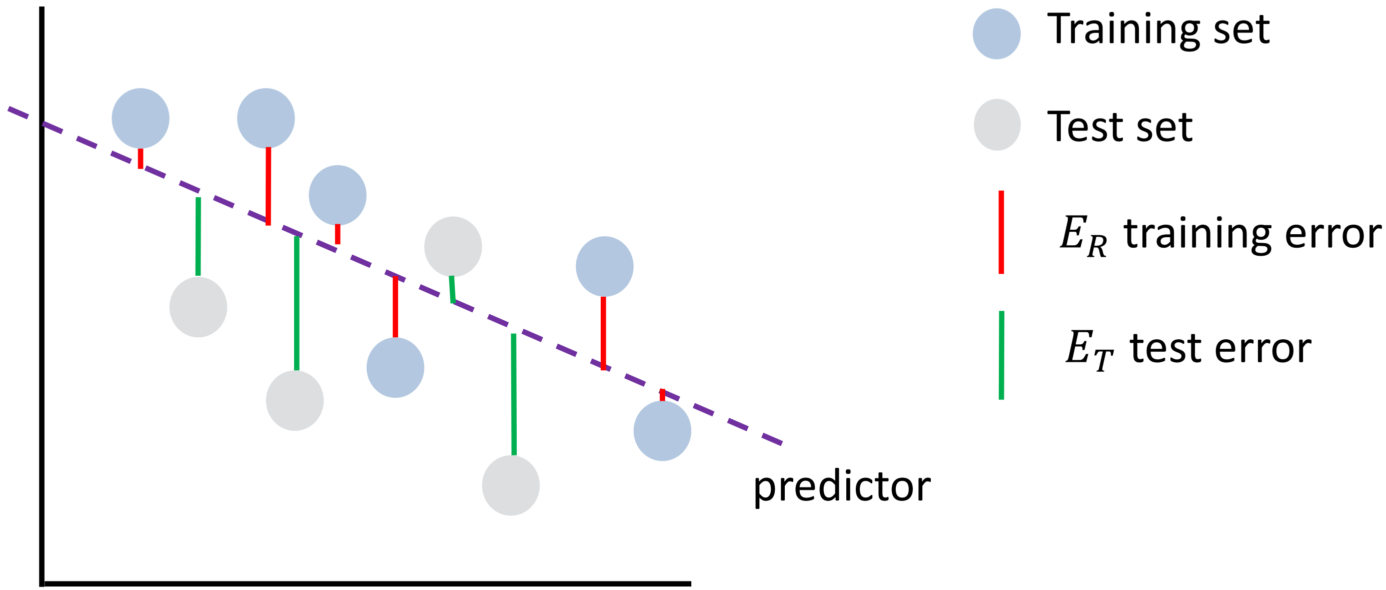


Train/Test Splits



Generally, the ML algorithm will select the predictor that minimizes the training error……
….hoping that it will then exhibit similar (hence, low) test error

Train/Test Splits



What can we expect for E_R and E_T?

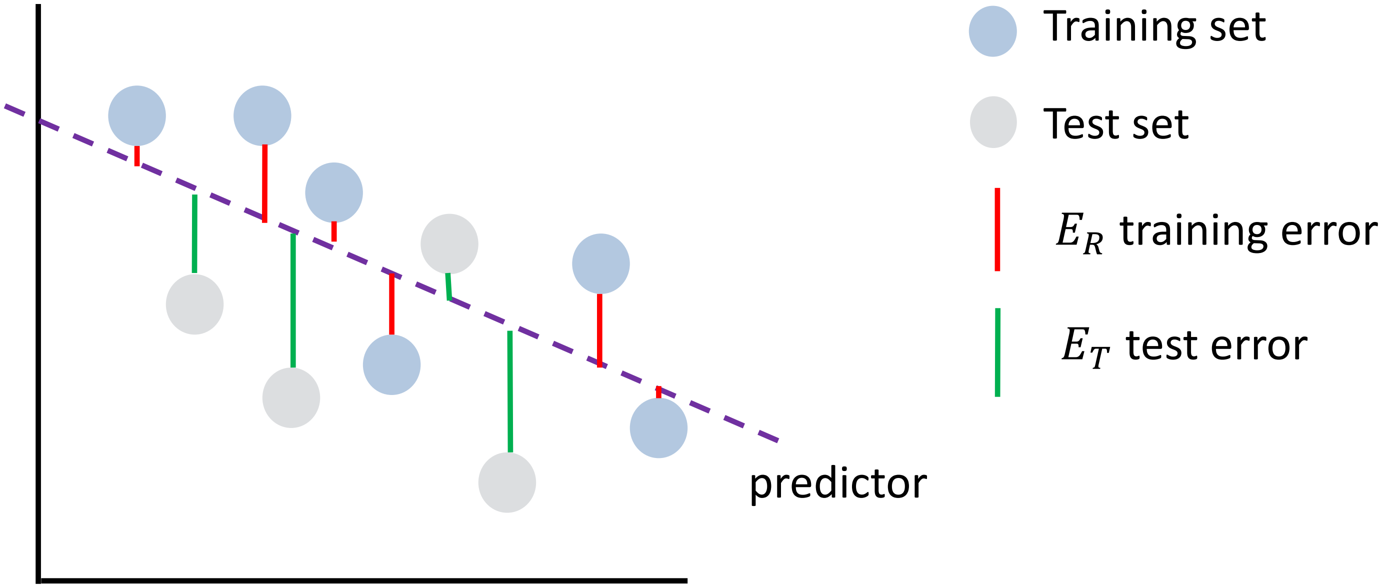
a) $E_R \geq E_T$

b) $E_T > E_R$

c) $|E_R - E_T| > 0$

d) E_R and E_T are roughly the same with some noise perturbation

Train/Test Splits



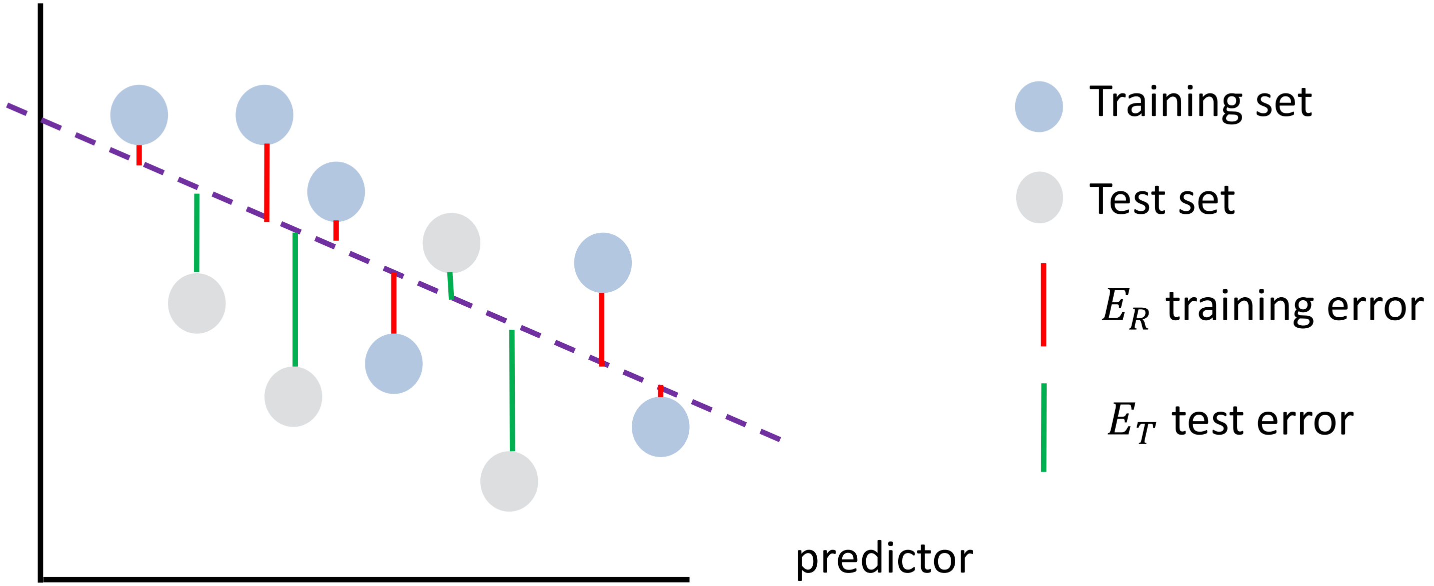
What can we expect for E_R and E_T?

- $E_R \geq E_T$
- $E_T > E_R$
- $|E_R - E_T| > 0$
- E_R and E_T are roughly the same with some noise perturbation

If your model isn't "right" yet (i.e. in practice, most of the time)

- We have not observed enough training points

Train/Test Splits



What can we expect for E_R and E_T?

- E_R and E_T are roughly the same with some noise perturbation

Once we observe enough training point and we have "learnt well"
Ideally we would like to bound the likelihood of observing errors

$$P(|E_T - E_R| > \epsilon) \leq \delta$$

In general, for good ML algorithms we have $E[E_T - E_R] = 0$

Generalization guarantees

- Generalization error:

$$E_G = |E_{train} - E_{test}|$$

- Statistical learning theory provides the tools to characterize the distribution of E_G

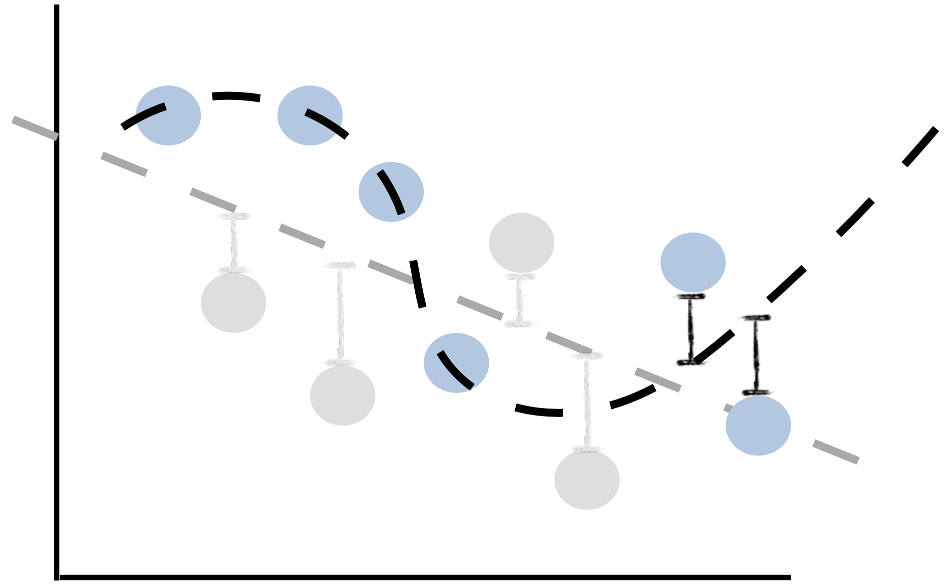
$$P(E_G > \epsilon) < \delta$$

- The distribution we can claim depends on parameters which capture the complexity of the class of models we are considering

Generalization guarantees

- These bounds are often rather pessimistic when compared with actual performance of ML algorithms
  - They state a much higher requirement for number of observations
  - They state much weaker guarantees than those observed
- Still, very important tool! We want to guarantee that something is going to work!

Complexity and overfitting



The more complex the possible models, the more likely we are to observe a large discrepancy between E_R and E_T

- The more complex the model, the more we tend to overfit to the training data
- In other words, we need more training samples to “learn well”

Overfitting

- Models are likely to overfit when the model is more "complex" than is needed to explain the variation we care about
- "Complex" generally tied to the number of parameters (i.e., features)
- When the number of parameters is >= the number of observations, you can trivially memorize your training data, often without learning anything generalizable to test time