Machine Learning in Three Lectures

Sach Mukherjee

DZNE, Bonn

What is machine learning?

- Machine learning (ML) is a subfield located at the intersection of computer science and computational statistics focused on making *decisions based on data*
- Notion of "decisions" or "actions" is very broad and includes many tasks currently done by humans hence close connection between ML and Artificial Intelligence (AI)

What is machine learning?

- ML is both a technology/tool and a conceptual approach. Now plays key – and often critical – role in many fields and sectors
- ML method = (model, estimator, computation), evaluated in context of specific goal. Can mix-and-match, may be trade-offs between three aspects
- Common thread is philosophy of learning from data and specific notion of empirical testing wrt problem of interest
- Note that ML $\neq$ AI $\neq$ "deep learning"

What is machine learning?

- Although ML makes much use of probability, optimization, linear algebra etc. its focus is very distinct and lies in systematically building on three key ideas:
  1. The generality of mappings: many practical problems can be viewed as requiring a mapping from one set to another
  2. Functions need not be specified upfront, but can be estimated ("learned") from data starting from essentially generic families.
  3. Generic regularization approaches can allow effective learning of flexible models. Statistical decision theory is critical

- Often the mathematics needed is surprisingly simple – ML is as much as a way of thinking as anything else
- Easiest way to get a feeling is to look at examples

Plan for the lectures

- Aim of the lectures is to give a self-contained introduction to some key ideas in ML
- Trade-off between presenting "unified" view and sticking to standard names for tasks/models, will lean towards the latter, but show connections throughout
- Sequence: first describe several concrete tasks, to fix ideas and give a feeling for the field, and later take a step back to a more general view

- Today: *supervised learning*

Supervised learning

Some notation

- Generic $n \times p$ data matrix $X = [x_1 \dots x_n]^{\mathrm{T}}$, understood as n observations in p dimensions
- Where there is an "input" and "output", usually x, y respectively
- $f(\cdot; \theta)$ refers to family of functions or distributions, parameterized by (possibly high-dimensional) θ
- Refer to θ as *parameter*, f as *model*

Some notation

- "Hat" indicates *estimate* from data, e.g. for unknown parameter θ, corresponding estimate is $\hat{\theta} = \hat{\theta}(X)$
- $\hat{\theta}(X)$ is a function of data, should be thought of as a random variable (RV)
- *Variance of estimator* means $\mathrm{Var}(\hat{\theta}(X))$ (depends on sample size n)
- *Likelihood* is the joint distribution of the data under some model $p(X \mid \theta)$ usually treated as a function of the parameter θ
- Warning: I will overload X, Y, meaning should be clear from context

Supervised learning

- *Supervised learning* is the base problem in machine learning, intuitive and extraordinarily useful

Supervised learning

- *Supervised learning* is the base problem in machine learning, intuitive and extraordinarily useful

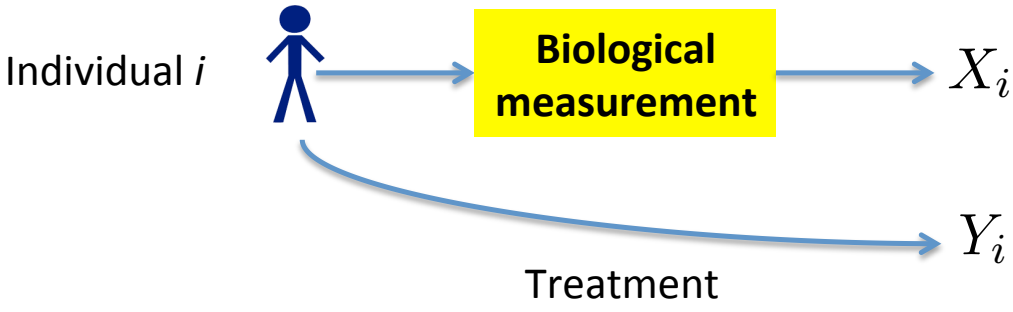
- Problem: want to predict some $y \in \mathcal{Y}$ from available inputs $x \in \mathcal{X}$
- Focus on
  $x \in \mathbb{R}^p$ and
  $y \in \{0, 1\}$ (*classification*) or
  $y \in \mathbb{R}$ (*regression*)
- Idea is to "learn" a function $\hat{f} : \mathcal{X} \to \mathcal{Y}$ using dataset $D_n = (x_i, y_i)_{i=1\ldots n}$ (i.e. with both x's and y's available in the data)
- $\hat{f}$ should be "good" in a certain sense that we will discuss later

Supervised learning

- *Supervised learning* is the base problem in machine learning, intuitive and extraordinarily useful

- Problem: want to predict some $y \in \mathcal{Y}$ from available inputs $x \in \mathcal{X}$
- Focus on
  $x \in \mathbb{R}^p$ and
  $y \in \{0, 1\}$ (*classification*) or
  $y \in \mathbb{R}$ (*regression*)
- Idea is to "learn" a function $\hat{f} : \mathcal{X} \to \mathcal{Y}$ using dataset $D_n = (x_i, y_i)_{i=1\ldots n}$ (i.e. with both x's and y's available in the data)
- $\hat{f}$ should be "good" in a certain sense that we will discuss later

- Called "supervised" because both x, y available at outset, hence like learning with a teacher
- Framework extremely general, x, y could be almost anything...



Input X_i

Output Y_i “Duck” “Tiger”

“Dear Mr Smith,

Congratulations! You have won a prize! We are delighted…

X_i

Spam Not spam Y_i

Classification

Classification

- Task: given data

$$(x_i, y_i)_{i=1\ldots n},\ x_i \in \mathbb{R}^p,\ y \in \{0, 1\},$$

  learn a function $\hat{f} : \mathbb{R}^p \to \{0, 1\}$

- $\hat{f}$ should be an accurate classifier in the sense that for a *new* pair (X', Y'), we would like $Pr(\hat{f}(X') = Y')$ to be high

LETTER

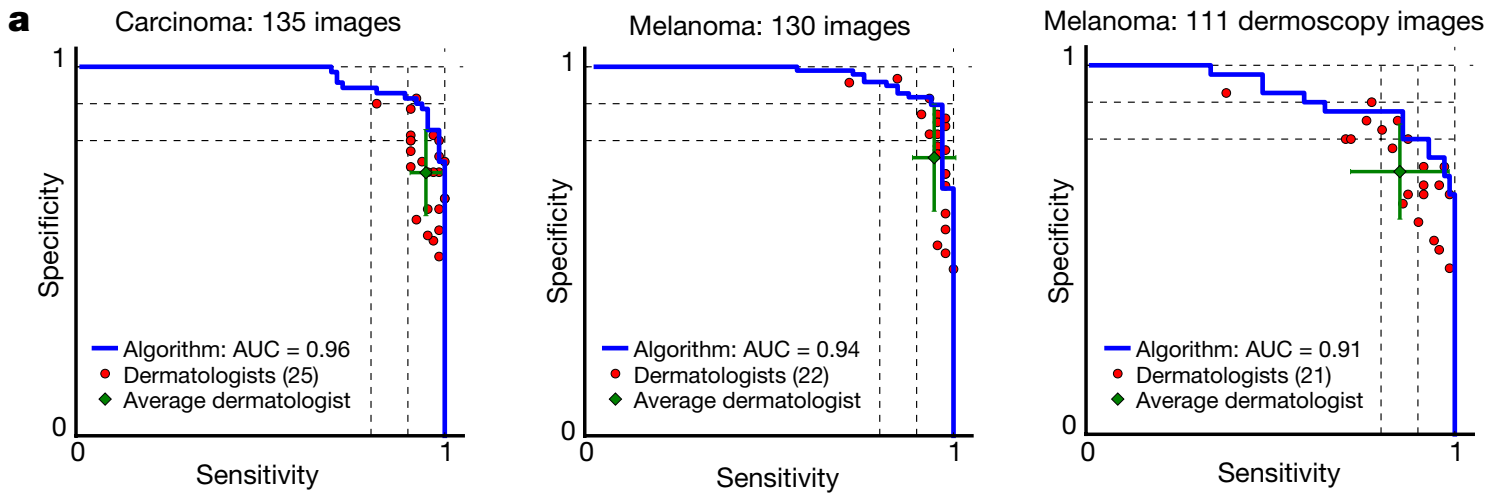
doi:10.1038/nature21056

Dermatologist-level classification of skin cancer with deep neural networks

Andre Esteva[1]*, Brett Kuprel[1]*, Roberto A. Novoa[2,3], Justin Ko[2], Susan M. Swetter[2,4], Helen M. Blau[5] & Sebastian Thrun[6]

Skin cancer, the most common human malignancy[1–3], is primarily diagnosed visually, beginning with an initial clinical screening and followed potentially by dermoscopic analysis, a biopsy and histopathological examination. Automated classification of skin

images (for example, smartphone images) exhibit variability in factors such as zoom, angle and lighting, making classification substantially more challenging[23,24]. We overcome this challenge by using a data-driven approach—1.41 million pre-training and training images



Output sets can be very complicated...

man in black shirt is playing guitar.

construction worker in orange safety vest is working on road.

(Karpathy & Li, CVPR, 2015)

A probability model for classification

- Treat X, Y as RVs and classify via

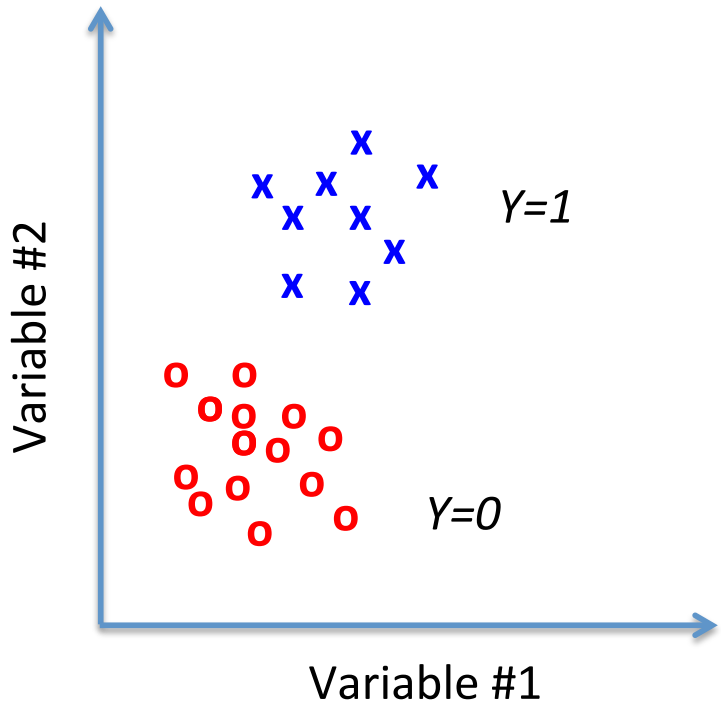
$$P(Y=1|X) = \frac{p(X|Y=1)P(Y=1)}{p(X|Y=1)P(Y=1) + p(X|Y=0)P(Y=0)}$$

- Classifier is

$$\hat{f}(X) = \underset{k \in \{0,1\}}{\operatorname{argmax}} \hat{P}(Y=k|X),$$

  with $\hat{P}$ being analogue of P estimated from data

- In practice, need a model for X, e.g. $X|Y=k \sim N(\mu_k, \Sigma_k)$


Variable #2
Variable #1
Y=1
Y=0

Gaussian class conditionals

- The boundary between the classes is

$$\hat{P}(Y = 1 \mid X) = \hat{P}(Y = 0 \mid X) = 1/2,$$

  known as the *decision boundary*
- For $X \mid Y{=}k \sim N(\mu_k, \Sigma_k)$ this is linear for $\Sigma_0 = \Sigma_1$ and quadratic otherwise

Logistic regression

- Assume $\Sigma_1 = \Sigma_0$ (linear boundary)
- A different view of the model comes from writing the class probability as

$$P(Y=1 \mid X=x) = \frac{p(X|Y=1)P(Y=1)}{\underbrace{p(X|Y=1)P(Y=1)}_{=p_1} + \underbrace{p(X|Y=0)P(Y=0)}_{=p_0}}$$

$$= \frac{1}{1+\frac{p_0}{p_1}}$$

$$= \frac{1}{1+\exp(-\beta^{\mathrm{T}}x+c)}$$

with β, c being unknown parameters

Logistic regression

- Assume $\Sigma_1 = \Sigma_0$ (linear boundary)
- A different view of the model comes from writing the class probability as

$$P(Y=1 \mid X=x) = \frac{p(X|Y=1)P(Y=1)}{\underbrace{p(X|Y=1)P(Y=1)}_{=p_1} + \underbrace{p(X|Y=0)P(Y=0)}_{=p_0}}$$

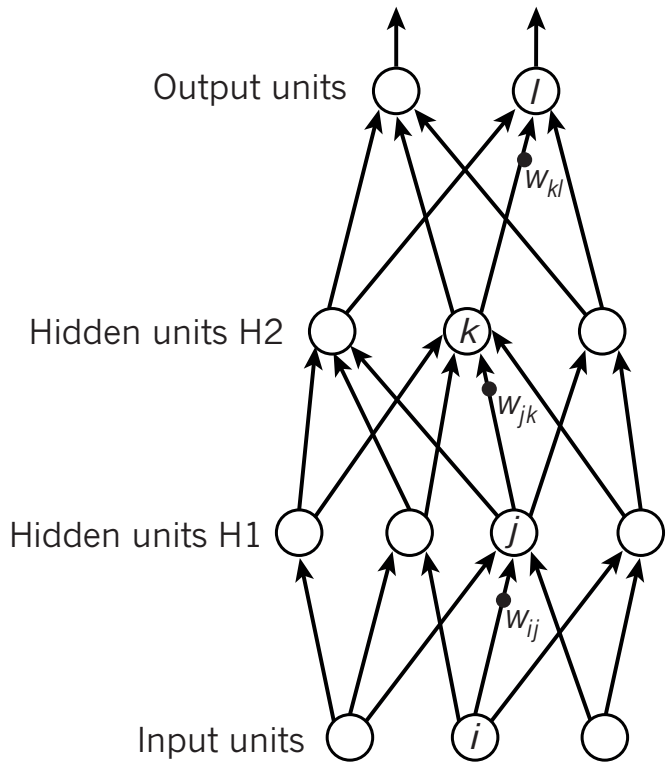
$$= \frac{1}{1+\frac{p_0}{p_1}}$$

$$= \frac{1}{1+\exp(-\beta^{\mathrm{T}}x + c)}$$

with β, c being unknown parameters

- This approach is called *logistic regression*, since it is a linear regression pushed through a logistic function
- The approach – of pushing a high-dimensional linear combination through a nonlinearity – is common in ML and is e.g. what happens in a single layer of a neural network

Neural network with hidden layers



(LeCun et al., Nature, 2015)

Generative vs. discriminative learning

- Consider the foregoing classification model. The parameters need to be set from observed data to obtain a useable classifier. How should the parameters be set?
- The discussion suggests two broad strategies: (i) treat as a density estimation problem, i.e. try to model the assumed data-generating process directly or (ii) optimize the logistic function to maximize prediction accuracy
- Two approaches are called *generative* and *discriminative* in the ML literature, reflecting the fact that modelling the (assumed) data-generating process and directly optimizing predictive ability are in general different goals

Regression

Regression

- Task: given data

$$(x_i, y_i)_{i=1\ldots n},\ x_i \in \mathbb{R}^p,\ y \in \mathbb{R},$$

  learn a function $\hat{f} : \mathbb{R}^p \to \mathbb{R}$

- For prediction, $\hat{f}$ should be accurate in the sense that for a *new* pair (x', y'), we would like $\hat{f}(x') \approx y'$

Linear regression

- Consider the linear model $f(x) = x^{\mathrm{T}}\beta$, with p-dimensional parameter β
- The parameters have to be fitted using the available data
- Let $X = [x_1 \dots x_n]^{\mathrm{T}}$ be a $n \times p$ data matrix and $Y = [y_1 \dots y_n]^{\mathrm{T}}$ an n-vector of corresponding outputs, then we want $X\beta \approx Y$

Linear regression

- One approach is to consider the optimization

$$\hat{\beta} = \underset{\beta}{\text{argmin}} \, \| Y - X\beta \|_2^2$$

- This is a least squares problem and has solution

$$\hat{\beta} = (X^{\mathrm{T}} X)^{-1} X^{\mathrm{T}} Y$$

- For probability model $Y \mid X{=}x, \beta \sim N(x^{\mathrm{T}}\beta, \sigma^2)$ the *maximum likelihood* solution is the same

Digression: expected loss (more later)

- A statistical decision is an action taken on the basis of data. Want to evaluate the quality of a candidate $\hat{f}$
- The *expected loss* or *risk* associated with $\hat{f}$ is

$$R(\hat{f}) = \mathbb{E}[L(Y, \hat{f}(X))]_{q(X,Y)}$$

  where q is the joint distribution over (X, Y) (this is unknown)

- Notice that this is *not* wrt the parameter, but wrt the prediction

Overfitting and higher dimensional models

- Consider a transformation of X via some $\Phi : \mathbb{R}^p \to \mathbb{R}^d$ and write $\Phi(X) = [\Phi(x_1) \dots \Phi(x_n)]^{\mathrm{T}}$ for the resulting $n \times d$ data matrix
- Model is

$$\hat{y}_i = \Phi(x_i)^{\mathrm{T}} \tilde{\beta}, \ \tilde{\beta} \in \mathbb{R}^d,$$

  solution as before, matrix $\Phi(X)$ replacing X

- Consider the transformed model for one dimensional x's and with Φ_k as the k^{th} order polynomial
- Fix number of data n and as before let $D_n = (x_i, y_i)_{i=1\dots n}$

Overfitting and higher dimensional models

- Consider expected loss

$$\begin{aligned} R &= \mathbb{E}[(Y - \hat{Y}(X))^2]_{q(X,Y)} \\ &= \mathbb{E}[(Y - \Phi_k(X)^{\mathrm{T}} \hat{\beta}(D_n))^2]_{q(X,Y)} \end{aligned}$$

- What happens to R as k gets larger, keeping n fixed?

Overfitting and higher dimensional models

- This phenomenon is known as *overfitting*
- Note that the sequence of models indexed by k are nested in the sense that the simpler models are always special cases
- That is, larger k means a strictly richer model class
- The issue arises due to the fact that as k gets larger the statistical variance increases and so does the risk
- Issue is central to real world ML – how many variables to include? How complex a model? More variables means a strictly richer model but at some point risk will increase (depends on the precise estimator)

Regularization for higher dimensional models

- Return to the linear model, recall the optimization

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \| Y - X\beta \|_2^2$$

- Obviously with $p > n$, this does not work and for p close to n will have poor behaviour (most modern applications have $p \gg n$)

Regularization for higher dimensional models

- Return to the linear model, recall the optimization

$$\hat{\beta} = \underset{\beta}{\text{argmin}} \, \| Y - X\beta \|_2^2$$

- Obviously with $p > n$, this does not work and for p close to n will have poor behaviour (most modern applications have $p \gg n$)
- Consider instead the constrained optimization

$$\hat{\beta} = \underset{\beta}{\text{argmin}} \, \| Y - X\beta \|_2^2 + \lambda \|\beta\|_2^2$$

- This can be viewed as "penalizing" extreme entries in β and has solution

$$\hat{\beta} = (X^{\mathrm{T}} X + \lambda I_p)^{-1} X^{\mathrm{T}} Y$$

- Known as *ridge regression*, simple but very effective in practice, widely used to build predictive models in high dimensions

Sparse high dimensional models

- Return (again!) to the linear model, recall the optimization

$$\hat{\beta} = \underset{\beta}{\text{argmin}} \, \| Y - X\beta \|_2^2$$

- For large p, an interesting question is whether some *subset* $A \subset \{1, \ldots, p\}$ of the variables might be sufficient
- If so, the sparsity would itself be a kind of regularization (lower effective model dimension) and it may be interesting to understand *which* variables are selected

Sparse high dimensional models: a direct approach

- A direct approach would be to explore subsets up to some maximum size c, i.e. to consider the constrained optimization

$$\hat{\beta} = \underset{\beta}{\text{argmin}} \, \| Y - X\beta \|_2^2, \;\; s.t. \| \beta \|_0 \leq c$$

- This would indeed regularize the problem and improve prediction performance
- But the problem is that the optimization is non-convex and the discrete model space is huge

Sparse high dimensional models: a convex approach

- Counting non-zeros is not a tractable approach
- Consider instead

$$\hat{\beta} = \underset{\beta}{\text{argmin}} \, \|Y - X\beta\|_2^2 + \lambda \|\beta\|_1$$

- This is convex and can be efficiently optimized for large p *but induces sparsity in* β
- So-called ℓ_1-penalized methods are widely used for many kinds of models in supervised learning and beyond

Digression: estimation in high-dimensions

- Unconstrained estimation in high dimensions does not scale!
- Nice example from classical density estimation
- Silverman (1986) computes n needed to achieve specified relative mean squared error wrt true density, for samples from a p-dimensional Gaussian
- For equivalent of $n = 4$ in one dimension, one needs $n = 842,000$ in $p = 10$ dimensions!

Why is learning in high dimensions possible?

- Even "big" data is essentially never enough to allow simple estimation of high-dimensional densities, coefficients etc.
- Historically, this set things back, because of a belief that nothing could be done for large p, so instead better to try to pre-select variables, focus the problem etc.

Why is learning in high dimensions possible?

- High-dimensional estimation now better understood, a main message is that properties of (regularized) estimators often better than expected, some classical results in a way too general/pessimistic
- In parallel, empirical evidence is that in many settings "high-dimensional-plus-regularization" can outperform "low-dimensional-and-unconstrained" in practice
- Essentially due to the fact that real data have (usually hidden) low-dimensional structure and it is possible to design regularization schemes which can themselves adapt efficiently to the data

Evaluating predictors

"With four parameters I can fit an elephant, and with five I can make him wiggle his trunk"

- Today, models with *millions* of free parameters are routinely used
- How should one evaluate a fitted predictor in practice?
- For regularized approaches, how should one guide the regularization?
- Often the regularization parameter λ can be thought of as controlling model complexity (e.g. level of sparsity)
- If the model is too simple, it may not be able to mimic the unknown underlying process but if too complex it will overfit
- First, consider what we would *like* to do but cannot

Expected loss

- The *expected loss* or *risk* associated with $\hat{f}$ is

$$R(\hat{f}) = \mathbb{E}[L(Y, \hat{f}(X))]_{q(X,Y)}$$

where q is the joint distribution over (X, Y) (this is unknown)

Expected loss

- The *expected loss* or *risk* associated with $\hat{f}$ is

$$R(\hat{f}) = \mathbb{E}[L(Y, \hat{f}(X))]_{q(X,Y)}$$

where q is the joint distribution over (X, Y) (this is unknown)

- Making initial data D_n explicit write

$$R(\hat{f}) = \mathbb{E}[L(Y, f(X; \hat{\theta}(D_n)))]_{q(X,Y)}$$

where $\hat{\theta}(D_n)$ is the estimated model parameter

Expected loss

- The *expected loss* or *risk* associated with $\hat{f}$ is

$$R(\hat{f}) = \mathbb{E}[L(Y, \hat{f}(X))]_{q(X,Y)}$$

where q is the joint distribution over (X, Y) (this is unknown)

- Making initial data D_n explicit write

$$R(\hat{f}) = \mathbb{E}[L(Y, f(X; \hat{\theta}(D_n)))]_{q(X,Y)}$$

where $\hat{\theta}(D_n)$ is the estimated model parameter

- This is arguably the right measure of how effective $\hat{f}$ is – notice how this addresses the "fitting an elephant" concern
- Unfortunately we don't have access to R

Empirical analogue: sample risk

- We don't have access to the *risk*

$$R(\hat{f}) = \mathbb{E}[L(Y, \hat{f}(X))]_{q(X,Y)},$$

but we do have a sample D_n from q. Why not use the sample analogue? I.e.:

$$\hat{R}(\hat{f}) = \frac{1}{n} \sum_i L(y_i, \hat{f}(x_i))$$

- But making initial data D_n explicit

$$\hat{R}(\hat{f}) = \frac{1}{n} \sum_i L(y_i, f(x_i; \hat{\theta}(D_n)))$$

makes the problem clear – *we are using the same data to build and evaluate the model!*

- This is another way of looking at over-fitting – the model can choose to fit the "noise" rather than the signal and this will show up as "good" performance

Empirical analogue: train and test

- This suggests a simple fix. Randomly split the data D_n into two halves D_{train} and D_{test}. Each is now a random sample (of size $n/2$) from q. Let the corresponding indices be $\mathcal{I}_{\text{train}}$ and $\mathcal{I}_{\text{test}}$ (i.e. these partition $\{1 \dots n\}$)
- Now consider the quantity

$$\hat{R}(\hat{f}) = (n/2)^{-1} \sum_{i \in \mathcal{I}_{\text{test}}} L(y_i, f(x_i; \hat{\theta}(D_{\text{train}})))$$

- Here the data used to fit ("train") the model are disjoint from those used to test it, hence any over-fitting should show up in $\hat{R}$

Empirical analogue: train and test

- This suggests a simple fix. Randomly split the data D_n into two halves D_{train} and D_{test}. Each is now a random sample (of size $n/2$) from q. Let the corresponding indices be $\mathcal{I}_{\text{train}}$ and $\mathcal{I}_{\text{test}}$ (i.e. these partition $\{1 \dots n\}$)
- Now consider the quantity

$$\hat{R}(\hat{f}) = (n/2)^{-1} \sum_{i \in \mathcal{I}_{\text{test}}} L(y_i, f(x_i; \hat{\theta}(D_{\text{train}})))$$

- Here the data used to fit ("train") the model are disjoint from those used to test it, hence any over-fitting should show up in $\hat{R}$
- Train/test is a core paradigm in ML. Key idea is to note that f is fit on (finite) D_n, but the quantity R we'd really like to minimize scores performance on unseen data – *fitting is not predicting*

Empirical analogue: cross validation

- Splitting the data into train and test is fine but note two things
  (1) Optimal regularization depends on n – more data means one can "afford" a richer model
  (2) The dataset size under splitting is in fact *halved*
- Hence this may lead to a *too simple* model

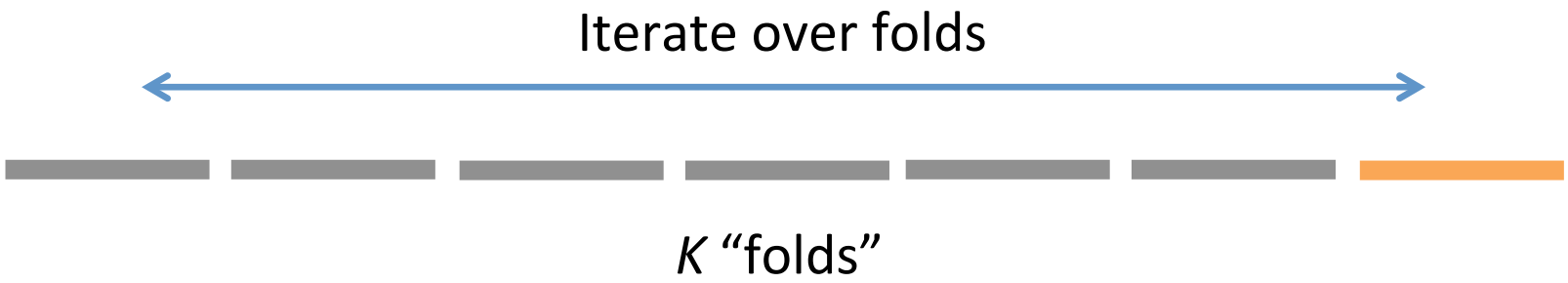
Empirical analogue: cross validation

- Splitting the data into train and test is fine but note two things
  (1) Optimal regularization depends on n – more data means one can "afford" a richer model
  (2) The dataset size under splitting is in fact *halved*
- Hence this may lead to a *too simple* model

- Alternative is to randomly split data into K same sized blocks, training on $K-1$ and testing on the left out one
- Iterate so that all data are used to test (and train)
- Then, the training sample is of size $\frac{K-1}{K}n$, i.e. closer to the n of interest

- This is called *K-fold cross-validation* and is the most widely-used empirical testing scheme in ML

K-fold cross validation



Learning curves

- More generally, the behaviour of training error and true risk R with n and model complexity d is central to ML
- *Learning curves* refer to plots of training and test error vs. n
- Asymptotes – wrt complexity and n – are conceptually important
- ML methods are geared towards negotiating the interplay between regularization and model complexity in light of available data and against behavior wrt the loss function.

Lecture I: summary

- Machine learning: a highly general approach to solving potentially complex problems
- Core ideas are simple, but a very specific mindset
- Regularization is essential for high-dimensional problems, has statistical and not just numerical effects
- Decision theory and empirical risk are critical in keeping track of whether models are effective or not and to guard against over- or under-fitting