Lecture 001

Statistical learning: Foundations

Edward Rubin
05 January 2021

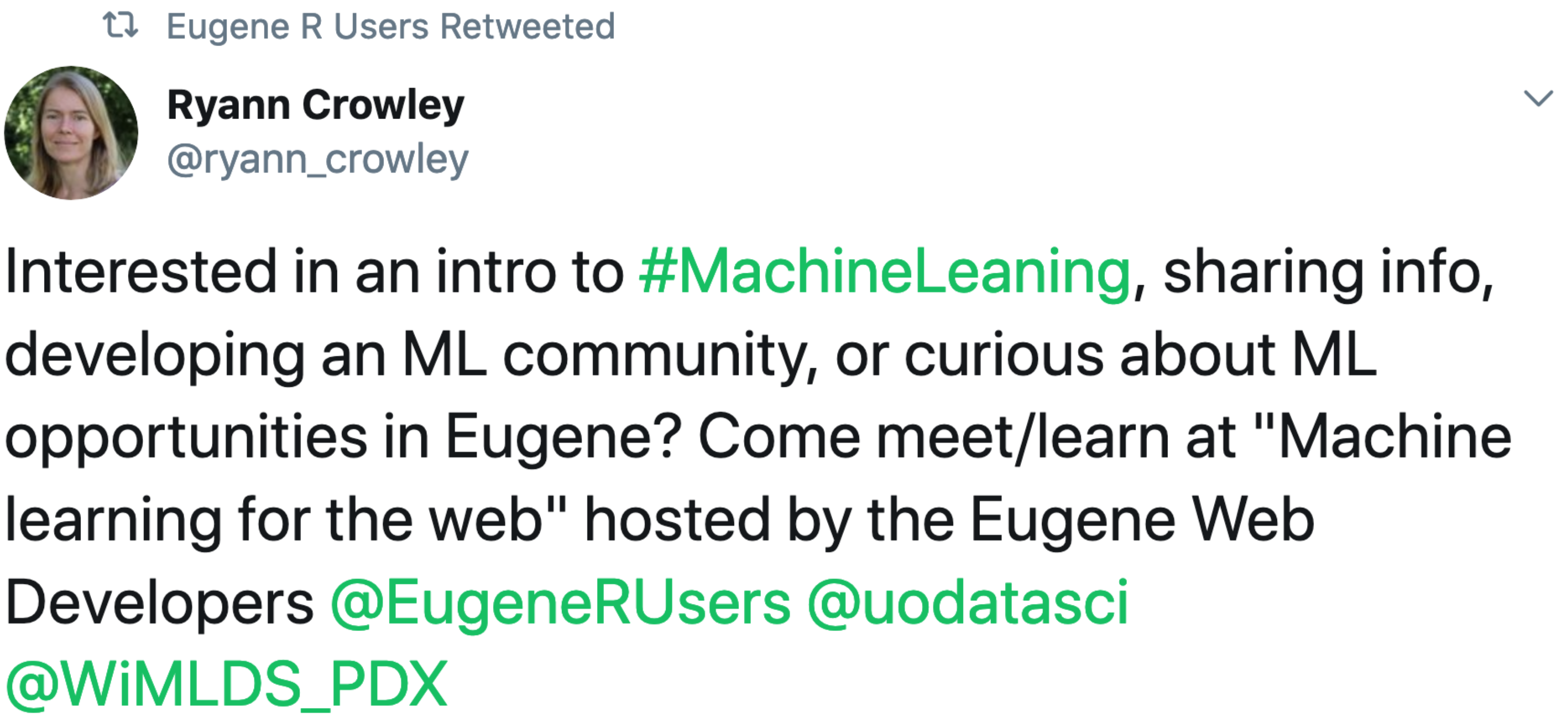
Admin

Admin

Today

In-class

- *Course website:* https://github.com/edrubin/EC524W21/
- *Resources*
    - RStudio cheatsheets, books, and tutorials
    - UO library
    - See course page for more...
- Formalizing statistical learning, notation, goals (and problems)



Eugene R Users Retweeted

Ryann Crowley
@ryann_crowley

Interested in an intro to #MachineLeaning, sharing info, developing an ML community, or curious about ML opportunities in Eugene? Come meet/learn at "Machine learning for the web" hosted by the Eugene Web Developers @EugeneRUsers @uodatasci @WiMLDS_PDX

Tweet; h/t: Grant McDermott

Admin

Upcoming

Readings

- *Today*
  - *ISL* Ch1–Ch2
  - Prediction Policy Problems by Kleinberg *et al.* (2015)
- *Next*
  - *ISL* Ch. 3–4

Problem set Soon.

Statistical learning

Statistical learning

What is it?

Statistical learning is a set of tools developed to understand/model data.

Examples

- **Regression analysis** quantifies the relationship between an outcome and a set of explanatory variables—most usefully in a causal setting.
- **Exploratory data analysis** (EDA) is a preliminary, often graphical, "exploration" of data to understand levels, variation, missingess, *etc.*
- **Classification trees** search through explanatory variables, splitting along the most "predictive" dimensions (random forests extend trees).
- **Regression trees** extend *classification trees* to numerical outcomes (random forests extend, as well).
- **K-means clustering** partitions observations into K groups (clusters) based upon a set of variables.

Statistical learning

What is it good for?

A lot of things. We tend to break statistical-learning into two(-ish) classes:

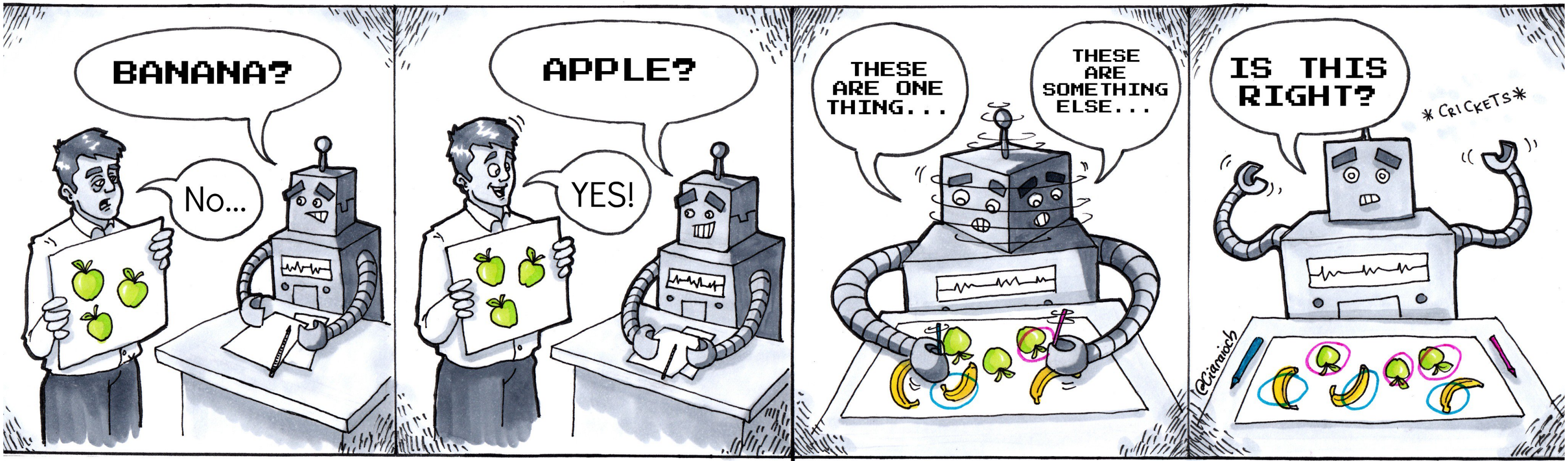
1. **Supervised learning** builds ("learns") a statistical model for predicting an **output** ($\mathbf{y}$) given a set of **inputs** ($\mathbf{x}_1, \ldots, \mathbf{x}_p$), *i.e.*, we want to build a model/function f

   $$\mathbf{y} = f(\mathbf{x}_1, \ldots, \mathbf{x}_p)$$

   that accurately describes $\mathbf{y}$ given some values of $\mathbf{x}_1, \ldots, x_p$.

2. **Unsupervised learning** learns relationships and structure using only **inputs** ($x_1, \ldots, x_p$) without any *supervising* output—letting the data "speak for itself."

Semi-supervised learning falls somewhere between these supervised and unsupervised learning—generally applied to supervised tasks when labeled **outputs** are incomplete.



Supervised Learning

Unsupervised Learning

Source

Statistical learning

Output

We tend to further break **supervised learning** into two groups, based upon the **output** (the outcome we want to predict):

1. **Classification tasks** for which the values of $\mathbf{y}$ are discrete categories
   E.g., race, sex, loan default, hazard, disease, flight status

2. **Regression tasks** in which $\mathbf{y}$ takes on continuous, numeric values.
   E.g., price, arrival time, number of emails, temperature

Note$_1$ The use of *regression* differs from our use of *linear regression*.

Note$_2$ Don't get tricked: Not all numbers represent continuous, numerical values—*e.g.*, zip codes, industry codes, social security numbers.[†]

[†] Q Where would you put responses to 5-item Likert scales?

Statistical learning

Why *Learning*?

Q What puts the "learning" in statistical/machine learning?

A Most learning models/algorithms will **tune model parameters** based upon the observed dataset—learning from the data.

Notation

Our class will typically follow the notation and definitions of *ISL*.

Notation

Data

n gives the number of observations

p represents the number of variables available for predictions

$\mathbf{X}$ is our $n \times p$ matrix of predictors

- *Other names* ***features****, inputs, independent/explanatory variables, …*
- $x_{i,j}$ is observation i (in $1, \ldots, n$) on variable j (for j in $1, \ldots, p$)

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,p} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,p} \end{bmatrix}$$

Notation

Dimensions of $\mathbf{X}$

Now let us split our $\mathbf{X}$ matrix of predictors by its two dimensions.

Observation i is a p-length vector

$$\boldsymbol{x}_i = \begin{bmatrix} x_{i,1} \\ x_{i,2} \\ \vdots \\ x_{i,p} \end{bmatrix}$$

Variable j is a n-length vector

$$\mathbf{x}_j = \begin{bmatrix} x_{1,j} \\ x_{2,j} \\ \vdots \\ x_{n,j} \end{bmatrix}$$

Applied to R:

- `dim(x_df)` = n p
- `nrow(x_df)` $= n$; `ncol(x_df)` $= p$
- `x_df[1,]` $(i = 1)$; `x_df[,1]` $(j = 1)$

Notation

Outcomes

In supervised settings, we will denote our **outcome variable** as $\mathbf{y}$.

Synonyms output, outcome, dependent/response variable, ...

The outcome for our ith obsevation is y_i. Together the n observations form

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

and our full dataset is composed of $\left\{ (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) \right\}$.

Back to the problem of (supervised) statistical learning…

Statistical learning

The goal

As defined before, we want to *learn* a model to understand our data.

1. Take our (numeric) output $\mathbf{y}$.
2. Imagine there is a function f that takes inputs $\mathbf{X} = \mathbf{x}_1, \ldots, \mathbf{x}_p$ and maps them, plus a random, mean-zero error term ε, to the output.

$$\mathbf{y} = f(\mathbf{X}) + \varepsilon$$

Q What is f?

A *ISL:* f represents the *systematic* information that $\mathbf{X}$ provides about $\mathbf{y}$.

Q How else can you describe f?

Statistical learning

Our missing f

$$\mathbf{y} = f(\mathbf{X}) + \varepsilon$$

Q f is unknown (as is ε). What should we do?

A Use the observed data to learn/estimate $f(\cdot)$, *i.e.*, construct $\hat{f}$.[†]

Q Okay. How?

A *How do I estimate f?* is one way to phrase *all questions* that underly statistical learning—model selection, cross validation, evaluation, *etc.*

All of the techniques, algorithms, tools of stat. learning attempt to accurately recover f based upon the settings' goals/limitations.

You'll have to wait on any real/specific answers...

[†] More notation: hats ($\hat{\ }$) are estimators/estimates.

Statistical learning

Learning from $\hat{f}$

There are two main reasons we want to learn about f

1. ***Causal* inference settings** How do changes in $\mathbf{X}$ affect $\mathbf{y}$?

   The territory of `EC523` and `EC525`.

2. **Prediction problems** Predict $\mathbf{y}$ using our estimated f, *i.e.*,

$$\hat{\mathbf{y}} = \hat{f}(\mathbf{X})$$

   our *black-box setting* where we care less about f than $\hat{\mathbf{y}}$.[†]

Similarly, in causal-inference settings, we don't particulary care about $\hat{\mathbf{y}}$.

[†] You shouldn't actually treat your prediction methods as total black boxes.

Statistical learning

Prediction errors

As tends to be the case in life, you will make errors in predicting $\mathbf{y}$.

The accuracy of $\hat{\mathbf{y}}$ depends upon **two errors**:

1. **Reducible error** The error due to $\hat{f}$ imperfectly estimating f.
   Reducible in the sense that we could improve $\hat{f}$.

2. **Irreducible error** The error component that is outside of the model f.
   Irreducible because we defined an error term ε unexplained by f.

Note As its name implies, you can't get rid of *irreducible* error—but we can try to get rid of *reducible* errors.

Statistical learning

Prediction errors

Why we're stuck with *irreducible* error

$$E\left[\{\mathbf{y} - \hat{\mathbf{y}}\}^2\right] = E\left[\left\{f(\mathbf{X}) + \varepsilon - \hat{f}(\mathbf{X})\right\}^2\right]$$

$$= \underbrace{\left[f(\mathbf{X}) - \hat{f}(\mathbf{X})\right]^2}_{\text{Reducible}} + \underbrace{\mathrm{Var}(\varepsilon)}_{\text{Irreducible}}$$

In less math:

- If ε exists, then $\mathbf{X}$ cannot perfectly explain $\mathbf{y}$.
- So even if $\hat{f} = f$, we still have irreducible error.

Thus, to form our **best predictors**, we will **minimize reducible error**.

Statistical learning

Which type of $\hat{f}$?

Once you have your inputs ($\mathbf{X}$) and output ($\mathbf{y}$) data, you still need to decide how parametric your $\hat{f}$ should be.[†]

Parametric methods assume a function typically involve two steps

1. Select a functional form (shape) to represent f
2. Train your selected model on your data $\mathbf{y}$ and $\mathbf{X}$.

Non-parametric methods avoid explicit assumption about the shape of f.
Attempt to flexibly fit the data, while trying to avoid overfitting.

[†] I'm saying "how parametric" b/c some methods are much more parametric than others.

Statistical learning

Which type of $\hat{f}$?

Methods' parametric assumptions come with tradeoffs.

Parametric methods

- + Simpler to estimate and interpret.
- – If assumed functional form is bad, model performance will suffer.

Non-parametric methods

- + Fewer assumptions. More flexibility.
- – Lower interpretability. Susceptible to overfitting. Want lots of data.

Example: Let's start with a pretty funky, nonlinear function.

Truth: The (nonlinear) $f(\mathbf{X})$ that we hope to recover.

The sample: $n = 70$ randomly drawn observations for $\mathbf{y} = f(\mathbf{x}_1,\, \mathbf{x}_2) + \varepsilon$

Estimated linear-regression model: $\hat{\mathbf{y}} = \hat{\beta}_0 + \hat{\beta}_1\mathbf{x}_1 + \hat{\beta}_2\mathbf{x}_2 + \hat{\beta}_3\mathbf{x}_1\mathbf{x}_2$

Prediction error from our fitted linear regression model

k-nearest neighbors (kNN) using k=5 (a *non-parametric* method)

k-nearest neighbors (kNN) using k=10 (notice increased smoothness)

k-nearest neighbors (kNN) using k=1 (notice decreased smoothness)

Prediction error from our fitted kNN (k=5) model

Prediction error from our fitted kNN (k=10) model

Prediction error from our fitted kNN (k=1) model

Recall **Prediction error** from our fitted linear regression model

Model accuracy

Questions

1. Which of the methods was the most flexible? Inflexible?

2. Why do you think kNN with k=1 had such low prediction error?

3. How could we (better) assess model/predictive performance?

4. Why would we ever want to choose a less flexible model?

Model accuracy

Measurement

You probably will not be surprised to know that there is no one-size-fits-all solution in statistical learning.

Q How do we choose between competing models?

A We're a few steps away, but before we do anything, we need a way to **define model performance**.

Model accuracy

Subtlety

Defining performance can actually be quite tricky...

Regression setting, 1 Which do you prefer?

1. Lots of little errors and a few really large errors.
2. Medium-sized errors for everyone.

Regression setting, 2 Is a 1-unit error (*e.g.*, $1,000) equally bad for everyone?

Model accuracy

Subtlety

Defining performance can actually be quite tricky...

Classification setting, 1 Which is worse?

1. False positive (*e.g.*, incorrectly diagnosing cancer)
2. False negative (*e.g.*, missing cancer)

Classification setting, 2 Which is more important?

1. True positive (*e.g.*, correct diagnosis of cancer)
2. True negative (*e.g.*, correct diagnosis of "no cancer")

Model accuracy

MSE

Mean squared error (MSE) is the most common[†] way to measure model performance in a regression setting.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} \left[y_i - \hat{f}\left(x_i \right) \right]^2$$

Recall: $y_i - \hat{f}\left(x_i \right) = y_i - \hat{y}_i$ is our prediction error.

Two notes about MSE

1. MSE will be (relatively) very small when **prediction error** is nearly zero.
2. MSE **penalizes** big errors more than little errors (the squared part).

† *Most common* does not mean best—it just means lots of people use it.

Model accuracy

Training or testing?

Low MSE (accurate performance) on the data that trained the model isn't actually impressive—maybe the model is just overfitting our data.[†]

What we want: How well does the model perform **on data it has never seen**?

This introduces an important distinction:

1. **Training data**: The observations (y_i, x_i) used to **train** our model $\hat{f}$.
2. **Testing data**: The observations (y_0, x_0) that our model has yet to see—and which we can use to evaluate the performance of $\hat{f}$.

Real goal: Low test-sample MSE (not the training MSE from before).

[†] Recall the kNN performance for k=1.

Next time: model performance, the variance-bias tradeoff, and kNN

Sources

These notes draw upon

- An Introduction to Statistical Learning (*ISL*)
  James, Witten, Hastie, and Tibshirani

- Python Data Science Handbook
  Jake VanderPlas

I pulled the comic from Twitter.

Table of contents