STAT 535 10/3/23

Lecture 2

- Prediction problems:
  classification, . . .
- ML terminology & Workflow
- Concepts in classification -
  real valued classifiers, ~~softmax~~

Recording -

QΦ feedback

Tutorial Predictors

Math !!

Tue 10/10
2 pm

LI

Lecture Notes I – Examples of Predictors

Marina Meilă
mmp@stat.washington.edu

Department of Statistics
University of Washington

September 25, 2023

Regression ✓

Prediction problems by the type of output

The "learning" paradigm and vocabulary

Classification concepts

The Nearest-Neighbor and kernel predictors

lecture

Linear predictors

Tutorial

- Least squares regression
- Linear Discriminant Analysis (LDA)
- QDA (Quadratic Discriminant Analysis)

NOT Linear!

- Logistic Regression
- ~~The PERCEPTRON algorithm~~

Classification and regression tree(s) (CART)

The Naive Bayes classifier

Hastie, Tibs, Friedman

Reading HTF Ch.: 2.3.1 Linear regression, 2.3.2 Nearest neighbor, 4.1–4 Linear classification, 6.1–3. Kernel regression, 6.6.2 kernel classifiers, 6.6.3 Naive Bayes, 9.2 CART, 11.3 Neural networks, Murphy Ch.: 1.4.2 nearest neighbors, 1.4.4 linear regression, 1.4.5 logistic regression, 3.5 and 10.2.1 Naive Bayes,4.2.1–3 linear and quadratic discriminant, 14.7.3– kernel regression, locally weighted regression, 16.2.1–4 CART, (16.5 neural nets), Bach Ch.:

Prediction problems by the type of output

In supervised learning, the problem is *predicting* the value of an **output** (or **response** – typically in regression, or **label** – typically in classification) variable Y from the values of some observed variables called **inputs** (or **predictors, features, attributes**) $(X_1, X_2, \ldots X_d) = X$. Typically we will consider that the input $X \in \mathbb{R}^d$. Prediction problems are classified by the type of response $Y \in \mathcal{Y}$:

- *regression*: $Y \in \mathbb{R}$
- *binary classification*: $\boxed{Y \in \{-1, +1\}}$
- *multiway classification*: $Y \in \{y_1, \ldots y_m\}$ a finite set
- *ranking*: $Y \in \mathbb{S}_p$ the set of permutations of p objects
- *multilabel classification* $Y \subseteq \{y_1, \ldots y_m\}$ a finite set (i.e. each X can have several labels)
- *structured prediction* $Y \in \Omega_V$ the state space of a graphical model over a set of [discrete] variables V

Ex:
news articles' topics

Binary classification (x^i, y^i)

$y^i = +1$ positive example

$y^i = -1$ negative —"—

Ex: x = paper submitted
y = accept/reject

Ex: Medical tests

Example ((Anomaly) detection.)

0 1

−1 +1

This is a binary classification problem. $Y \in \{\text{normal}, \text{abnormal}\}$. For instance, Y could be "HIV positive" vs "HIV negative" (which could be abbreviated as "+", "-") and the inputs X are concentration of various reagents and lymph cells in the blood.
Anomaly detection is a problem also in artificial systems, as any device may be functioning normally or not. There are also more general detection problems, where the object detected is of scientific interest rather than an "alarm": detecting Gamma-ray bursts in astronomy, detecting meteorites in Antarctica (a robot collects rocks lying on the ice and determines if the rock is terrestrial or meteorite). More recently, *Artificial Intelligence* tasks like detecting faces/cars/people in images or video streams have become possible.

Ex
- infectious disease
- engine state

Imagenet
MNIST

Example (**Multiway classification.**)

Handwritten digit classification: $Y \in \{0, 1, \ldots 9\}$ and X=black/white $64\times$ 64 image of the digit. For example, ZIP codes are being now read automatically off envelopes.
OCR (Optical character recognition). The task is to recognized printed characters automatically. X is again a B/W digital image, $Y \in \{a - z, A - Z, 0 - 9, ".", ",", \ldots\}$, or another character set (e.g. Chinese).

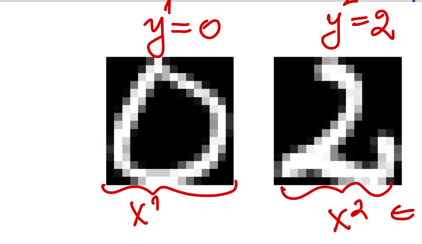
Example (**Diagnosis**)

Diagnosis is multiway classification + anomaly detection. $Y = 0$ means "normal/healthy", while $Y \in \{1, 2, \ldots\}$ enumerates failure modes/diseases.

"r = r = r"

$|y| = r > 2$ $\quad y \in \{1, \ldots r\}$

$\{0, \ldots r-1\}$



y = car states
$y = 1$ engine
2 = A/C, electric
3 = body
. . .

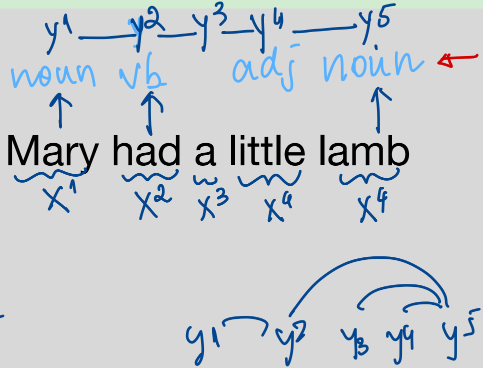
Large Language Models

Example (**Structured prediction.**)

Speech recognition. X is a segment of digitally recorded speech, Y is the word corresponding to it. Note that it is not trivial to *segment speech*, i.e to separate the speech segment that corresponds to a given word. These segments have different lengths too (and the length varies even when the same word is spoken).

The classification problem is to associate to each segment X of speech the corresponding word. But one notices that the words are not indepedent of other neighboring words. In fact, people speak in sentences, so it is natural to recognize each word in dependence from the others. Thus, one imposes a graphical model structure on the words corresponding to an utterance $X^1, X^2, \ldots X^m$. For instance, the labels $Y^{1:m}$ could form a *chain* $Y^1 - Y^2 - \ldots Y^m$. Other more complex graphical models structures can be used too.

= Dependency structure in y variables = graphical model



← depends on context

Prediction problems by the type of output

In supervised learning, the problem is *predicting* the value of an **output** (or **response** – typically in regression, or **label** – typically in classification) variable Y from the values of some observed variables called **inputs** (or **predictors, features, attributes**) $(X_1, X_2, \ldots X_d) = X$. Typically we will consider that the input $X \in \mathbb{R}^d$. Prediction problems are classified by the type of response $Y \in \mathcal{Y}$:

- *regression*: $Y \in \mathbb{R}$ ✓
- *binary classification*: $Y \in \{-1, +1\}$ ✓
- *multiway classification*: $Y \in \{y_1, \ldots y_m\}$ a finite set ✓
- *ranking*: $Y \in \mathbb{S}_p$ the set of permutations of p objects
- *multilabel classification* $Y \subseteq \{y_1, \ldots y_m\}$ a finite set (i.e. each X can have several labels) ✓
- ✓ *structured prediction* $Y \in \Omega_V$ the state space of a graphical model over a set of [discrete] variables V

input $X^1, \ldots X^P$ a set of objects (inputs)

[Q = query for search engines]

output y = ranking (=permutation) of $\{X^{1:P}\}$

$y = (X^{[1]}, X^{[2]}, \ldots)$

[most relevant

Pb: learning to rank

$\{X^{1:P}\} \xrightarrow{f} (X^{[1]} \ldots X^{[P]})$

The "learning" paradigm and vocabulary

- **predictor** = a [deterministic] function that associates to an input x a corresponding $\hat{y} = f(x)$.
- A predictor is a kind of model (not yet a statistical model, though).
- ~~**model**~~ **class** $\mathcal{F}$ = the set of possible predictors for a problem

predictor class

Sometimes called "models" too

abuse

- $\mathcal{D} = \{(x^1, y^1) \ldots (x^n, y^n)\}$ data, training set

Goal: given $\mathcal{D}$, given $\mathcal{F}$ $\}$ → "learn" $\hat{f} \in \mathcal{F}$ best

learned predictor

Non-probabilistic

Probabilistic

Ex: logistic regression $\hat{P}_{Y=1|X}$

In reality $y \sim P_{Y|X=x}$

model

"wreck a nice beach"

recognize speech

$\mathcal{F} = \{P_{Y|X}\}$ model class

Goal given $\mathcal{D}$, given $\mathcal{F}$ $\}$ → learn $\hat{P}_{Y|X} \in \mathcal{F}$ $\xrightarrow{\text{rule}}$ f

model

predictor

The "learning" paradigm and vocabulary

- **predictor** = a [deterministic] function that associates to an input x a corresponding $\hat{y} = f(x)$.
- A predictor is a kind of model (not yet a statistical model, though).
- **model class** $\mathcal{F}$ = the set of possible predictors for a problem

- **Training**
  - choose the "best" predictor in $\mathcal{F}$ (for a particular task)
  - based on a **sample** or (**training set**) of **labeled data**

$$\mathcal{D} = \{(x^1, y^1),\ (x^2, y^2),\ \ldots\ (x^n, y^n)\}$$

  - n is the **sample size**.
  - (x^i, y^i) are **examples**
  - In binary classification labels are conventionally in $\{\pm\}$ (or $\{\pm 1\}$). We use the terms **negative**, respectively **positive** examples
- **Prediction** (also called **testing**)
  - Given predictor f, and new input x, calculate

$$\hat{y} \;=\; f(x)$$

Prediction – the workflow

Training phase

- Get labeled data $\mathcal{D} = \{(x^1, y^1), (x^2, y^2), \ldots (x^n, y^n)\}$ 1.
- Choose model class $\mathcal{F}$ ← Model selection 2.
- Learn/estimate/fit the model $f \in \mathcal{F}$ from data $\mathcal{D}$ 3.
  - Here the goal is to find f that predicts $y^{1,\cdots N}$ well
  - How to do it is the **learning algorithm** and depends on $\mathcal{F}$ ⇒ $\hat{f}$

Ex: cross-validation

[**Validation phase** How good really is this f?] ← testing before $\hat{f}$ released output

e.g Model selection

"Testing" phase=Prediction

- now you have a predictor f, use it ← User I
- whenever new, unlabeled x comes in, output $\hat{y} = f(x)$

new input

$P[y \text{ wrong}]$ over $\mathcal{X}$

II ML developer (of new methods)

Testing = show (the world) that $\hat{f}$ is good

"test error"

Test set $\mathcal{D}^{test} = \{(\tilde{x}^1, \tilde{y}^1), \ldots (\tilde{x}^{\tilde{n}}, \tilde{y}^{\tilde{n}})\}$

Prediction – the workflow

Training phase

- Get labeled data $\mathcal{D} = \{(x^1, y^1), (x^2, y^2), \ldots (x^n, y^n)\}$
- Choose model class $\mathcal{F}$
- Learn/estimate/fit the model $f \in \mathcal{F}$ from data $\mathcal{D}$
  - Here the goal is to find f that predicts $y^{1,\ldots N}$ well
  - How to do it is the **learning algorithm** and depends on $\mathcal{F}$

[**Validation phase** How good really is this f?]

Learning Theory

- How to guarantee statistically that f predicts new $y(x)$ well

"Testing" phase=Prediction

- now you have a predictor f, use it
- whenever new, unlabeled x comes in, output $\hat{y} = f(x)$

Prediction problems by the type of output ✓

The "learning" paradigm and vocabulary ✓

· Classification concepts ←

The Nearest-Neighbor and kernel predictors

Linear predictors
- Least squares regression
- Linear Discriminant Analysis (LDA)
- QDA (Quadratic Discriminant Analysis)
- Logistic Regression
- The PERCEPTRON algorithm

Classification and regression tree(s) (CART)

The Naive Bayes classifier

Reading HTF Ch.: 2.3.1 Linear regression, 2.3.2 Nearest neighbor, 4.1–4 Linear classification, 6.1–3. Kernel regression, 6.6.2 kernel classifiers, 6.6.3 Naive Bayes, 9.2 CART, 11.3 Neural networks, Murphy Ch.: 1.4.2 nearest neighbors, 1.4.4 linear regression, 1.4.5 logistic regression, 3.5 and 10.2.1 Naive Bayes,4.2.1–3 linear and quadratic discriminant, 14.7.3– kernel regression, locally weighted regression, 16.2.1–4 CART, (16.5 neural nets), Bach Ch.:

The "sign trick" for transforming a regressor into a classifier

The *sign* function $\mathrm{sgn}(y) = y/|y|$ if $y \neq 0$ and 0 iff $y = 0$ turns a real valued variable Y into a discrete-valued one. This function is used to allow one to construct *real-valued classifiers*. In these classifiers, the model $f(x)$ is a real-valued function, and the prediction $\hat{y}$ is given by $\mathrm{sgn}(f(x))$.

Note that in a vanishingly small fraction of cases, when the value of $f(x)$ is exactly 0, no label will be assigned to the input x.

Classifiers with $\mathbb{R}$-valued outputs

Binary $y \in \{\pm 1\}$ $f : \mathcal{X} \to \{\pm 1\}$ $\mathcal{F} = \{ \quad \}$

but often

$$f : \mathcal{X} \to \mathbb{R}$$

$$\hat{y} = \operatorname{sign} f(x) \in \{\pm 1\}$$ ← $\hat{y}$ can be 0 (in practice $\hat{y} \neq 0$ almost always)

$$\operatorname{sign} z = \begin{cases} 1, & \text{if } z > 0 \\ -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \end{cases}$$

Ex: Logistic regression [Naive Bayes]

$$f(x) \equiv P[y = 1 \mid x] \in [0, 1]$$

$$\hat{y} = \operatorname{sign}\left(f(x) - \tfrac{1}{2}\right) = \begin{cases} 1 & \text{if } f(x) > \tfrac{1}{2} \\ -1 & < \tfrac{1}{2} \\ 0 & f(x) = \tfrac{1}{2} \end{cases}$$

Why?

- $f(x)$ models $P_{Y|X=x}$
- easier training [if $f(x) \in \{\pm 1\}$ gradient ∇f can't be used]
- $|f(x)|$ measures confidence in $\hat{y}$ value Ex: SVM

Real-valued outputs in multi-way classification

$y \in \{1, \dots r\}$

$f(x) \in \mathbb{R}^r \sim$ models $P_{Y|X=x}$ as categorical distribution

<u>Ex</u> $P[Y=1|X=x] = \dfrac{f_1(x)}{f_1(x) + \cdots f_r(x)}$ assuming $f_k(x) \geq 0$

Ex: digit classification

Softmax function ← (need not assume $f_k \geq 0$)

$$\left[\ \sigma_k(z_1, \dots z_r) = \frac{e^{z_k}}{e^{z_1} + \dots e^{z_r}} \qquad k = 1:r\ \right] \Rightarrow \sigma_{1:r} \begin{matrix} > 0 \\ < 1 \end{matrix}, \quad \sigma_1 + \dots \sigma_r = 1$$

for any $z = \begin{bmatrix} z_1 \\ \vdots \\ z_r \end{bmatrix}$

$$r = 2 \Rightarrow z = \begin{bmatrix} z_0 \\ z_1 \end{bmatrix} \Rightarrow \sigma_1 = \frac{e^{z_1}}{e^{z_0} + e^{z_1}}$$ logistic function

$$\hat{y} = \underset{k=1:r}{\arg\max}\ \sigma_k(f(x)) = \underset{k=1:r}{\arg\max}\ f_k(x)$$