MS&E 125: Intro to Applied Statistics

Train, Test, Validate

Professor Udell

Management Science and Engineering
Stanford

March 23, 2023

Exercise

You run a hospital. A vendor wants to sell you a new machine learning system for diagnosing colon cancer. The vendor tells you the system works with 99% accuracy.

You stand to save millions of dollars and hundreds of lives by catching the disease early. What evidence would you ask for to verify this claim?

ideas:

Exercise

You run a hospital. A vendor wants to sell you a new machine learning system for diagnosing colon cancer. The vendor tells you the system works with 99% accuracy.

You stand to save millions of dollars and hundreds of lives by catching the disease early. What evidence would you ask for to verify this claim?

ideas:

- what data was the system trained on?
- how well does it perform on your hospital's data?
- how well does it do on population subgroups?
- how many false positives / false negatives?

Generalization of supervised learning

- unknown target function $f : \mathcal{X} \to \mathcal{Y}$
- training examples $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$
- hypothesis set $\mathcal{H}$
- learning algorithm $\mathcal{A}$
- final hypothesis $g : \mathcal{X} \to \mathcal{Y}$

how well will our classifier do on **new** data?

A cautionary tale

- predict who will be approved for credit card
- training examples $\mathcal{D}$ = all previous applicants + decisions
- hypothesis set $\mathcal{H}$ = rules of the form

$$h(\text{applicant}) = \begin{cases} 1 & \text{applicant name} = \text{Abigail Adams} \\ 1 & \text{applicant name} = \text{Bhavik Balakrishnan} \\ 1 & \text{applicant name} = \text{Carrie Chen} \\ 1 & \dots \\ -1 & \text{otherwise} \end{cases}$$

A cautionary tale

- predict who will be approved for credit card
- training examples $\mathcal{D}$ = all previous applicants + decisions
- hypothesis set $\mathcal{H}$ = rules of the form

$$h(\text{applicant}) = \begin{cases} 1 & \text{applicant name} = \text{Abigail Adams} \\ 1 & \text{applicant name} = \text{Bhavik Balakrishnan} \\ 1 & \text{applicant name} = \text{Carrie Chen} \\ 1 & \dots \\ -1 & \text{otherwise} \end{cases}$$

- learning algorithm $\mathcal{A}$: pick h that performs best on $\mathcal{D}$

final hypothesis h classifies X% of the training set correctly

A. 0
B. 50
C. 100

A cautionary tale

- predict who will be approved for credit card
- training examples $\mathcal{D}$ = all previous applicants + decisions
- hypothesis set $\mathcal{H}$ = rules of the form

$$h(\text{applicant}) = \begin{cases} 1 & \text{applicant name} = \text{Abigail Adams} \\ 1 & \text{applicant name} = \text{Bhavik Balakrishnan} \\ 1 & \text{applicant name} = \text{Carrie Chen} \\ 1 & \dots \\ -1 & \text{otherwise} \end{cases}$$

- learning algorithm $\mathcal{A}$: pick h that performs best on $\mathcal{D}$
- final hypothesis **memorizes** the data

$$h(\text{applicant}) = \begin{cases} 1 & \text{applicant} = x_i \text{ and } y_i = 1 \text{ for some } i \\ -1 & \text{otherwise} \end{cases}$$

A cautionary tale

- predict who will be approved for credit card
- training examples $\mathcal{D}$ = all previous applicants + decisions
- hypothesis set $\mathcal{H}$ = rules of the form

$$h(\text{applicant}) = \begin{cases} 1 & \text{applicant name} = \text{Abigail Adams} \\ 1 & \text{applicant name} = \text{Bhavik Balakrishnan} \\ 1 & \text{applicant name} = \text{Carrie Chen} \\ 1 & \dots \\ -1 & \text{otherwise} \end{cases}$$

- learning algorithm $\mathcal{A}$: pick h that performs best on $\mathcal{D}$
- final hypothesis **memorizes** the data

$$h(\text{applicant}) = \begin{cases} 1 & \text{applicant} = x_i \text{ and } y_i = 1 \text{ for some } i \\ -1 & \text{otherwise} \end{cases}$$

how well will our classifier do on **new** data?

Estimate performance

how well will our model do on **new** data?

A. terribly
B. ok
C. extremely well
D. can't tell

Estimate performance

how well will our model do on **new** data?

A. terribly
B. ok
C. extremely well
D. can't tell

how to predict performance on new data?:

Estimate performance

how well will our model do on **new** data?

A. terribly
B. ok
C. extremely well
D. can't tell

how to predict performance on new data?:

- (given infinite data) evaluate model on new data
- (given finite data) split data; fit on one part, evaluate on another part

Error metric

how to measure how well model fits?
define an **error metric** $\mathbf{error} : \mathcal{Y} \times \mathcal{Y} \to \mathbf{R}$

$$\mathbf{error}(y, y') = \text{penalty for predicting } y' \text{ when true label is } y$$

choose an error metric that makes sense for your application!

Error metric: examples

Regression.

- **Square error.** $\textbf{error}_{\text{sq}}(y, y') = (y - y')^2$

Error metric: examples

Regression.

- **Square error.** $\textbf{error}_{\mathsf{sq}}(y, y') = (y - y')^2$
- **Absolute error.** $\textbf{error}_{\mathsf{abs}}(y, y') = |y - y'|$

Error metric: examples

Regression.

- **Square error.** $\text{error}_{\text{sq}}(y, y') = (y - y')^2$
- **Absolute error.** $\text{error}_{\text{abs}}(y, y') = |y - y'|$

Classification.

- **Misclassification error.** $\text{error}_{\text{mis}}(y, y') = \mathbb{1}(y \neq y')$

Error metric: examples

Regression.

- **Square error.** $\textbf{error}_{\text{sq}}(y, y') = (y - y')^2$
- **Absolute error.** $\textbf{error}_{\text{abs}}(y, y') = |y - y'|$

Classification.

- **Misclassification error.** $\textbf{error}_{\text{mis}}(y, y') = \mathbb{1}(y \neq y')$
- **Weighted misclassification error.** If false positives are β times worse than false negatives,

$$\textbf{error}_{\beta}(y, y') = \beta \mathbb{1}(y' = 1, y = -1) + \mathbb{1}(y' = -1, y = 1)$$

Error metric: examples

Regression.

- **Square error.** $\textbf{error}_{\text{sq}}(y, y') = (y - y')^2$
- **Absolute error.** $\textbf{error}_{\text{abs}}(y, y') = |y - y'|$

Classification.

- **Misclassification error.** $\textbf{error}_{\text{mis}}(y, y') = \mathbb{1}(y \neq y')$
- **Weighted misclassification error.** If false positives are β times worse than false negatives,

$$\textbf{error}_{\beta}(y, y') = \beta \mathbb{1}(y' = 1, y = -1) + \mathbb{1}(y' = -1, y = 1)$$

- **Balanced error rate.** If dataset contains n_+ examples with positive label and n_- negative,

$$\textbf{error}_{\text{bal}}(y, y') = \mathbb{1}(y \neq y') / (n_+ \mathbb{1}(y) + n_-(1 - \mathbb{1}(y)))$$

How to evaluate classification model

- precision
- recall
- false positive / false negative
- F1 score
- ROC, AUC, or AUROC
  - remove dependence on thresholds
- ... multiclass? ...

in sklearn...

Train and test error

- partition data $\mathcal{D}$ into
  - **training set** $\mathcal{D}_{\text{train}}$ and
  - **test set** $\mathcal{D}_{\text{test}}$

  so $\mathcal{D}_{\text{train}} \cap \mathcal{D}_{\text{test}} = \emptyset$, $\mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{test}} = \mathcal{D}$
- algorithm $\mathcal{A}$ is **only** allowed to see the training set
- **training error.**

$$E_{\text{train}}(g) \quad = \quad \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{i \in \mathcal{D}_{\text{train}}} \textbf{error}(y_i, g(x_i))$$

- **test error.**

$$E_{\text{test}}(g) \quad = \quad \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{i \in \mathcal{D}_{\text{test}}} \textbf{error}(y_i, g(x_i))$$

Train and test error

- partition data $\mathcal{D}$ into
  - **training set** $\mathcal{D}_{\text{train}}$ and
  - **test set** $\mathcal{D}_{\text{test}}$

  so $\mathcal{D}_{\text{train}} \cap \mathcal{D}_{\text{test}} = \emptyset$, $\mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{test}} = \mathcal{D}$
- algorithm $\mathcal{A}$ is **only** allowed to see the training set
- **training error.**

$$E_{\text{train}}(g) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{i \in \mathcal{D}_{\text{train}}} \textbf{error}(y_i, g(x_i))$$

- **test error.**

$$E_{\text{test}}(g) = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{i \in \mathcal{D}_{\text{test}}} \textbf{error}(y_i, g(x_i))$$

normalization $\implies$ metrics are called "mean square error" (MSE), "mean absolute error" (MAE), etc.

Generalization and Overfitting

- goal of model is **not** to predict well on $\mathcal{D}$
- goal of model is to predict well **on new data**

if the model has ____ training set error and ____ test set error, we say the model:

	low test set error	high test set error
low training set error	generalizes	overfits
high training set error	?!?!	underfits

Poll

How to fix underfitting?

A. use more complex model
B. use less complex model
C. add new features
D. remove features
E. find more data

Poll

How to fix overfitting?

A. use more complex model
B. use less complex model
C. add new features
D. remove features
E. find more data

Poll

Is it possible to overfit and underfit at the same time?

A. yes
B. no

how to choose train and test sets?

- at random (eg, toss random coin with prob p)
- more data in the training set improves the model fit
- more data in test set helps determine how well the model will work out of sample
- rule of thumb: put about 20% of data into the test set

how to choose train and test sets?

- at random (eg, toss random coin with prob p)
- more data in the training set improves the model fit
- more data in test set helps determine how well the model will work out of sample
- rule of thumb: put about 20% of data into the test set

Q: Consider a manufacturing application. Suppose your model will be trained on historical order volume data to predict future order volume. How should you choose your test set?

how to choose train and test sets?

- at random (eg, toss random coin with prob p)
- more data in the training set improves the model fit
- more data in test set helps determine how well the model will work out of sample
- rule of thumb: put about 20% of data into the test set

Q: Consider a manufacturing application. Suppose your model will be trained on historical order volume data to predict future order volume. How should you choose your test set?
A: Truncated timeseries. Backtesting. Extrapolation vs interpolation.

Validation

training set error improves with model complexity. how to decide which model to use? hold out a **validation set**

a simple and effective validation procedure:

- split data into training set $\mathcal{D}_{\text{train}}$ and test set $\mathcal{D}_{\text{valid}}$
- pick m different interesting model classes
  e.g., different ϕs: ϕ_1, ϕ_2, ..., ϕ_m
- fit ("train") models on training set $\mathcal{D}_{\text{train}}$
  get one model $h : \mathcal{X} \to \mathcal{Y}$ for each ϕs, and set

$$\mathcal{H} = \{h_1, h_2, \ldots, h_m\}$$

- compute error of each model on validation set $\mathcal{D}_{\text{valid}}$ and choose lowest:

$$g = \operatorname*{argmin}_{h \in \mathcal{H}} E_{\mathcal{D}_{\text{valid}}}(h)$$

Demo: Linear models (validation)

https://github.com/mse-125/demos

Cross-validation

a simple and effective procedure:

- pick a bunch of interesting model classes (*e.g.*, different ϕs)
- for each possible split of data into training set $\mathcal{D}$ and test set $\mathcal{D}'$
  - fit ("train") models on training set $\mathcal{D}$
  - compute error of each model on validation set $\mathcal{D}'$
- estimate error as average of error on each $\mathcal{D}'$

How to pick splits

how to pick splits?

- **Leave-one-out cross-validation:** $\mathcal{D}'$ has one example, $\mathcal{D}$ has all the rest
  - advantage: accurate
  - disadvantage: slow
- ***n*-fold cross-validation:** $\mathcal{D}'$ has $\frac{1}{n}$ of the examples, $\mathcal{D}$ has all the rest (usually, $n = 5$ or 10)
  - decently accurate, not too slow

Recap: Validation

- we care how well model performs on **new** data
- to simulate new data, split $\mathcal{D}$ into train and test set
- evaluate error on each via error metric
- if training set error is high, we say model fits poorly
  - solution: use more complex model, or add new features
- if test set error is much higher than training set error, we say model overfits
  - solution: make less complex model, or remove features