CS 581 – Advanced Artificial Intelligence

Topic: Learning – Naïve Bayes

Mustafa Bilgic

http://www.cs.iit.edu/~mbilgic

https://twitter.com/bilgicm

Classification

- AI → ML → Supervised Learning → Classification
- Given a dataset $D = \{\langle X_i, y_i \rangle\}$ where
  - X_i is the input
  - y_i is the discrete-valued output
- Learn a function $f(X_j) \rightarrow y_j$
- We would like f to **generalize** to **unseen** data
  - As opposed to memorize the given data

Classification Examples

- Email classification
- Medical diagnosis
- Face recognition
- Optical character/digit recognition
- Sentiment classification
- …

Algorithms

- Decision trees
- Nearest neighbor classification
- Naïve Bayes
- Logistic regression
- Support vector machines
- Neural networks
- …

Generalization

- The purpose of *f*
  - Is not to memorize the "seen" data
  - Is to generalize to "unseen" data
- We need a performance metric
- We need to test *f*'s performance on a dataset that it has not seen

Types of Errors – Classification

- Assume a target/positive class
  - Spam, HasHeartDisease, etc.
- *False positive*
  - Falsely classifying an object as positive
    - E.g., classifying a legitimate email as spam, diagnosing a healthy patient as having heart disease, and so on
  - Also called *Type I* error
- *False negative*
  - Falsely classifying an object as negative
    - E.g., classifying a spam email as not-spam, claiming that a heart-disease patient is healthy, and so on
  - Also called *Type II* error

A Few Performance Measures

- 0/1 loss; error or accuracy
- Precision
- Recall
- F1
- Log-loss
- Domain-specific performance measures,
- …

Confusion Matrix

		Predicted Class	
		Positive	**Negative**
Actual Class	**Positive**	True Positive	False Negative
	Negative	False Positive	True Negative

Accuracy

		Predicted Class	
		Positive	**Negative**
Actual Class	**Positive**	True Positive	False Negative
	Negative	False Positive	True Negative

$$Accuracy = \frac{Num\ Correct}{Data\ Size} = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision

		Predicted Class	
		Positive	**Negative**
Actual Class	**Positive**	True Positive	False Negative
	Negative	False Positive	True Negative

$$Precision = \frac{True\ Positive}{Predicted\ Positive} = \frac{TP}{TP + FP}$$

True Positive Rate – Recall – Sensitivity

		Predicted Class	
		Positive	**Negative**
Actual Class	**Positive**	True Positive	False Negative
	Negative	False Positive	True Negative

$$TPR = Recall = \frac{True\ Positive}{Actual\ Positive} = \frac{TP}{TP + FN}$$

True Negative Rate – Specificity

		Predicted Class	
		Positive	**Negative**
Actual Class	**Positive**	True Positive	False Negative
	Negative	False Positive	True Negative

$$TNR = Specificity = \frac{True\ Negative}{Actual\ Negative} = \frac{TN}{TN + FP}$$

False Positive Rate – Fall-out

		Predicted Class	
		Positive	**Negative**
Actual Class	**Positive**	True Positive	False Negative
	Negative	False Positive	True Negative

$$FPR = FallOut = \frac{False\ Positive}{Actual\ Negative} = \frac{FP}{TN + FP}$$

False Negative Rate – Miss Rate

		Predicted Class	
		Positive	**Negative**
Actual Class	**Positive**	True Positive	False Negative
	Negative	False Positive	True Negative

$$FNR = Miss\ Rate = \frac{False\ Negative}{Actual\ Positive} = \frac{FN}{TP + FN}$$

F1

		Predicted Class	
		Positive	**Negative**
Actual Class	**Positive**	True Positive	False Negative
	Negative	False Positive	True Negative

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

Splitting the dataset

1. Train-test splits
2. Train-validation-test splits
3. Cross-validation

Train-Test Split

- Randomly split the data into two disjoint sets
- A typical approach: 2/3 for train and 1/3 for test
- Train your model on training data and evaluate it on the test data
  - Use your favorite performance metric
- Report your performance as the expected performance on unseen data
- Caveats:
  - You need a large dataset for this to work
  - You cannot tune your parameters on the test data

Train-Validation-Test Split

- Split your data into three disjoint sets
  - Train, validation, test
- Train your model(s) on the training data
- Evaluate your model(s) on the validation data
- Pick the model that performs best on the validation data
- Test the model on the test data, and report its performance
- Caveat:
  - You need a really big dataset for this to work

Cross-Validation

- Split your data into k disjoint sets
- Each time, one set is the test set and the rest is the training set
- See OneNote for more detailed explanation and illustration

Real Life Measures

- Not as clean as the ones we discussed
- Imagine self-driving cars, medical diagnosis, crime prediction, fraud detection, and so on
- Usually, there is not a single performance measure
- Performance is handled on a case-by-case basis; not on an aggregate level

Naïve Bayes

Bayes Classifier

- Input: $\vec{X} = \langle X_1, X_2, \ldots, X_n \rangle$
- Output: Y
- Bayes classifier

$$P(Y \mid \vec{X}) = \frac{P(\vec{X} \mid Y)P(Y)}{P(\vec{X})} = \frac{P(Y)P(X_1, X_2, \ldots, X_n \mid Y)}{P(X_1, X_2, \ldots, X_n)}$$

$$P(X_1, X_2, \ldots, X_n) = \sum_y P(Y = y)P(X_1, X_2, \ldots, X_n \mid Y = y)$$

Assuming all variables are binary, how many independent parameters are needed for the Bayes classifier?

Bayes Classifier

- Assume a binary classification task, where the label Y is *spam* or ~*spam*
- Assume P(*spam*) = 0.4
- If you have seen an email X a times as *spam* and b times as ~*spam*
  - Using an MLE estimate for P(X|Y), what is P(Y|X)?
  - Using an LS estimate for P(X|Y), what is P(Y|X)?
  - What happens when either or both of a and b are zero?

Naïve Bayes Assumption

$$X_i \perp X_j \mid Y$$

Naïve Bayes

Bayes rule:

$$P(Y \mid X_1, X_2, \ldots, X_n) = \frac{P(Y)P(X_1, X_2, \ldots, X_n \mid Y)}{\sum_y P(y)P(X_1, X_2, \ldots, X_n \mid y)}$$

Assuming $X_i \perp X_j \mid Y$,
naïve Bayes:

$$P(Y \mid X_1, X_2, \ldots, X_n) = \frac{P(Y)\prod P(X_i \mid Y)}{\sum_y P(y)\prod P(X_i \mid y)}$$

Assuming all variables are binary, how many independent parameters are needed for the naive Bayes classifier?

Example

- See OneNote

Naïve Bayes Implementations

- Bernoulli / categorical naïve Bayes
  - Features are assumed to be binary / categorical
- Multinomial naïve Bayes
  - $P(\vec{X} \mid y)$ is a multinomial distribution
- Gaussian naïve Bayes
  - Each $p(x_i \mid y)$ is a Gaussian distribution

Reading

- http://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf
- https://en.wikipedia.org/wiki/Naive_Bayes_classifier
- https://scikit-learn.org/stable/modules/naive_bayes.html

Zero Probabilities

- Assume feature Xi is T for a particular object. Further,
- Assume $P(X_i = T|yes) = 0$ and $P(X_i = T|no) > 0$ and $P(X_j \mid yes) > 0$ and $P(X_j \mid no) > 0$ for all other features
  - What is $P(yes \mid \vec{X})$?
- Assume $P(X_i = T|yes) = 0$ and $P(X_i = T|no) = 0$ and $P(X_j \mid yes) > 0$ and $P(X_j \mid no) > 0$ for all other features
  - What is $P(yes \mid \vec{X})$?
- One solution: use LS for the parameter estimates

Multiplying Several Probability Numbers

- Assume we have 10,000 features
- What is $0.9^{10,000}$ using a computer?
- Try math.pow(0.9, 10000) in Python
- In Naïve Bayes,
  - $a = P(Y = T) \prod P(X_i | Y = T)$
  - $b = P(Y = F) \prod P(X_i | Y = F)$
  - $P\left(Y = T \middle| \vec{X}\right) = \frac{a}{a+b}$
  - If $a = b = 0$ in your code, then what?