Learning from Unlabeled Data

INFO-4604, Applied Machine Learning

University of Colorado Boulder

Nov 30, 2020

Prof. Abe Handler

Types of Learning

Recall the definitions of:

- Supervised learning
  - Most of the semester has been supervised
- Unsupervised learning
  - Example: k-means clustering
- Semi-supervised learning
  - More similar to supervised learning
    - Task is still to predict labels
    - But makes use of unlabeled data in addition to labeled
  - We haven't seen any algorithms yet

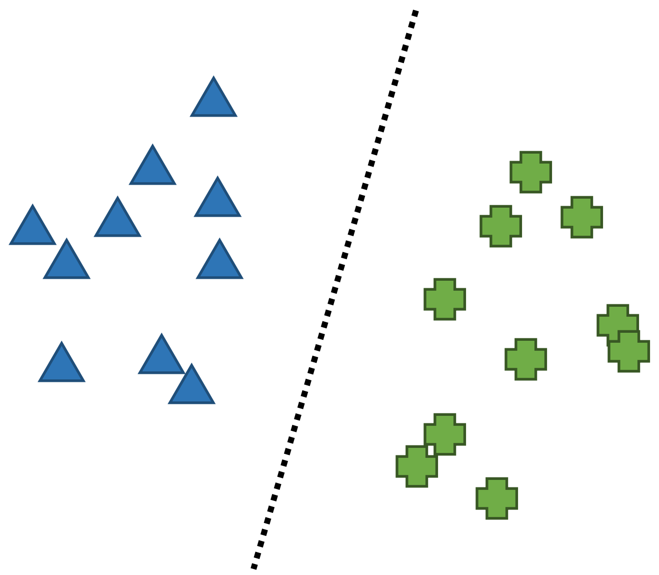
This Week

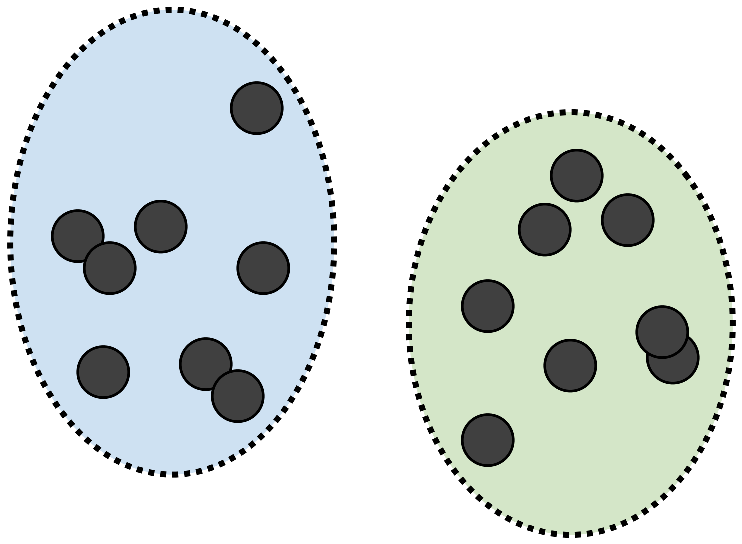
Semi-supervised learning

- General principles
- General-purpose algorithms
- Algorithms for generative models

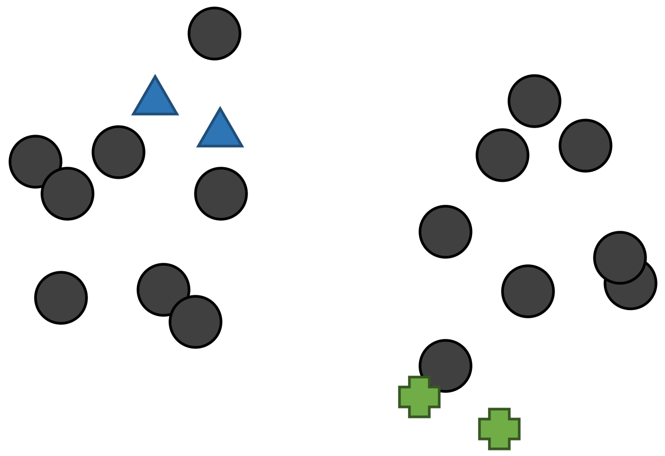
We'll also get into how these ideas can be applied to unsupervised learning as well

Types of Learning


Supervised learning

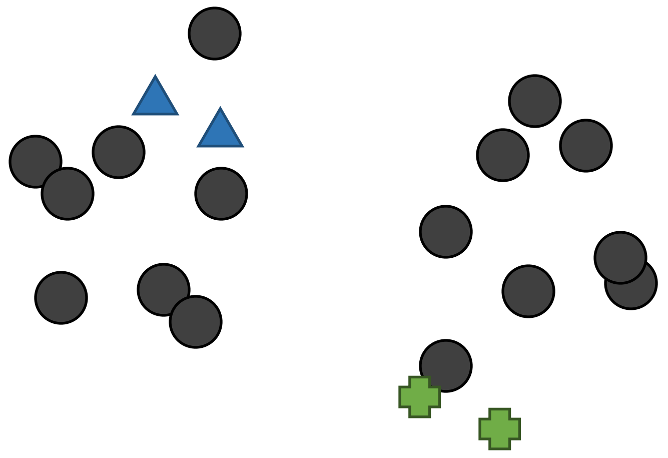

Unsupervised learning

Types of Learning



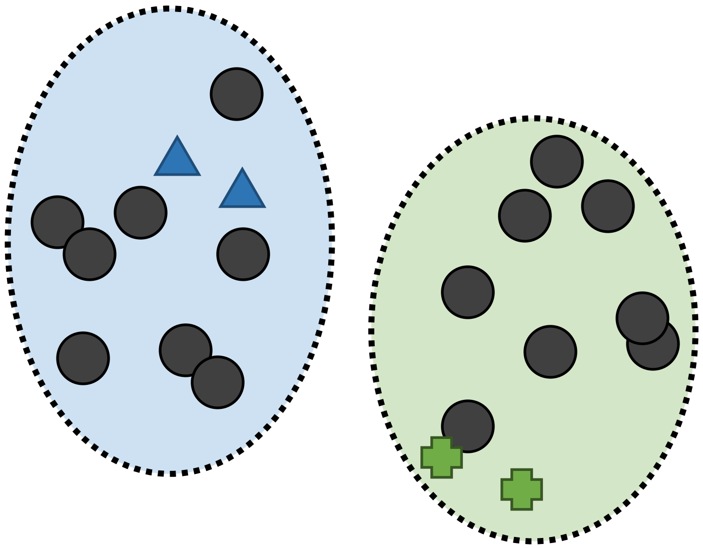
Semi-supervised learning

Types of Learning



Can combine supervised and unsupervised learning

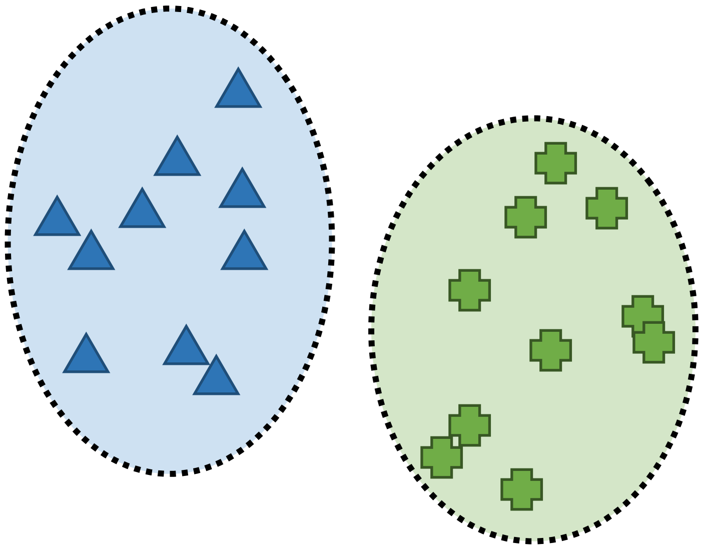
Types of Learning



Can combine supervised and unsupervised learning

- Two natural clusters

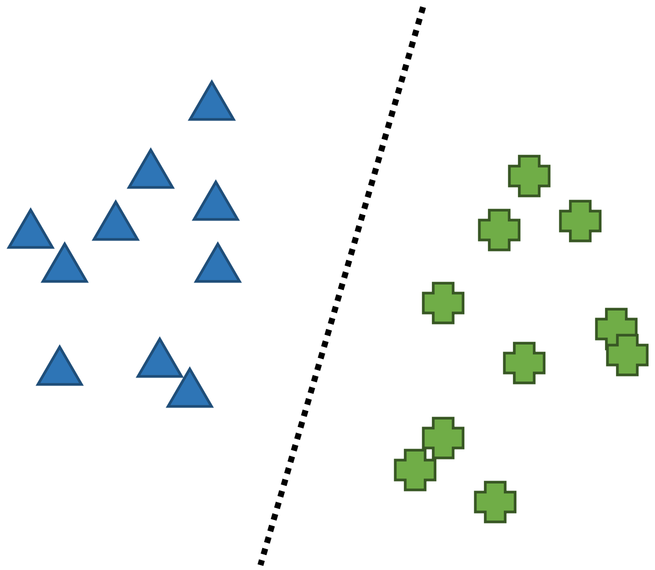
Types of Learning



Can combine supervised and unsupervised learning

- Two natural clusters
- Idea: assume instances within cluster share a label

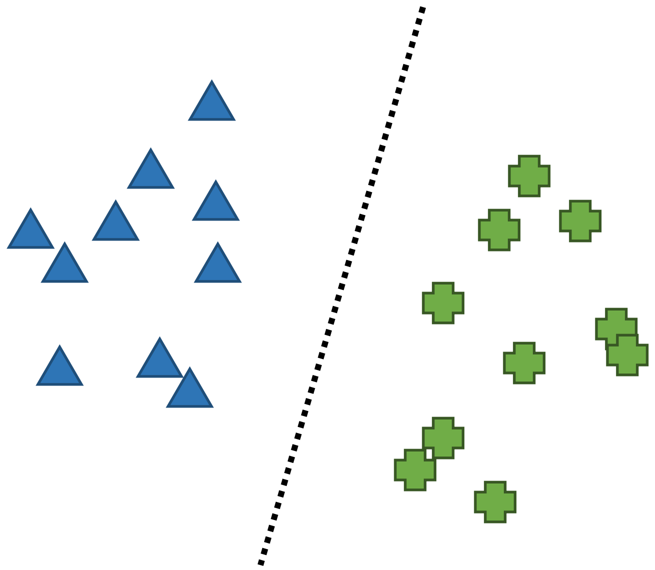
Types of Learning



Can combine supervised and unsupervised learning

- Two natural clusters
- Idea: assume instances within cluster share a label
- Then train a classifier on those labels

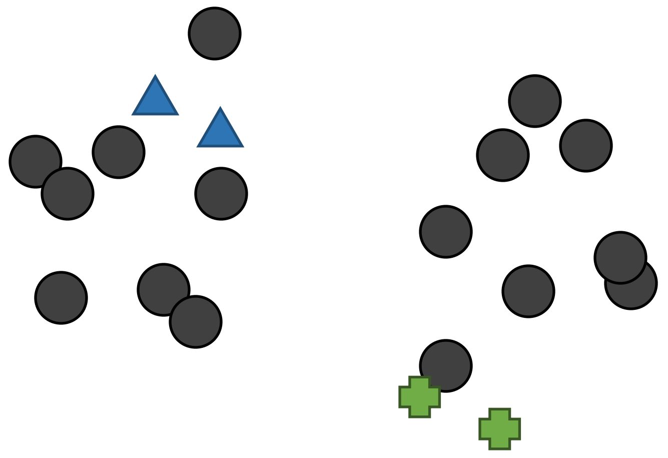
Types of Learning



This particular process is not a common method (though it is a valid one!)

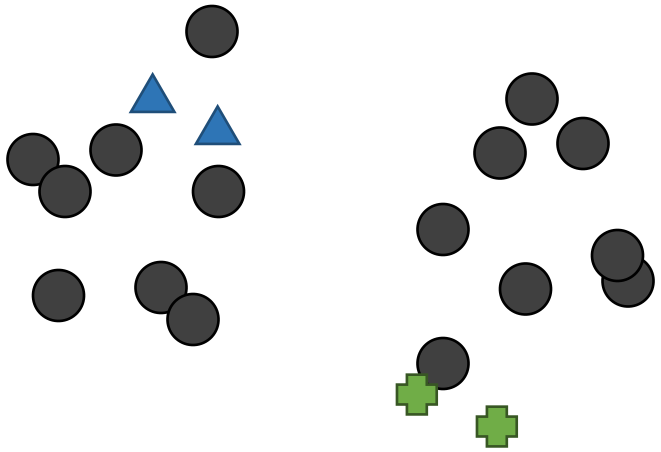
But it illustrates the ideas of semi-supervised learning

Types of Learning



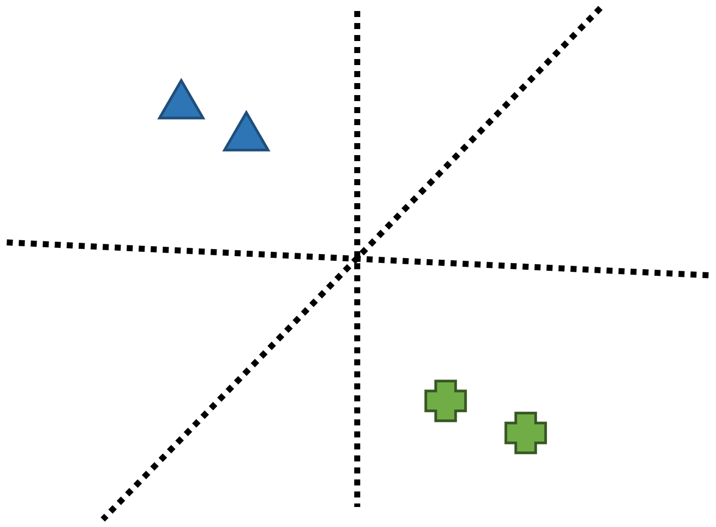
Semi-supervised learning

Types of Learning



Let's look at another illustration of *why* semi-supervised learning is useful

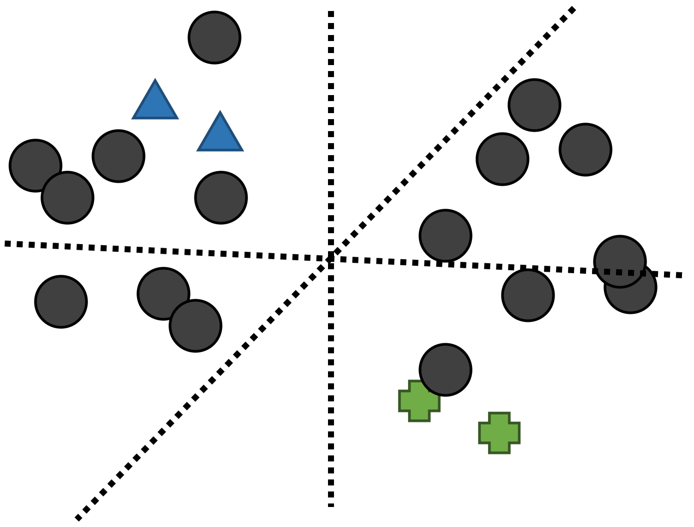
Types of Learning



If we ignore the unlabeled data, there are many hyperplanes that are a good fit to the training data

Types of Learning

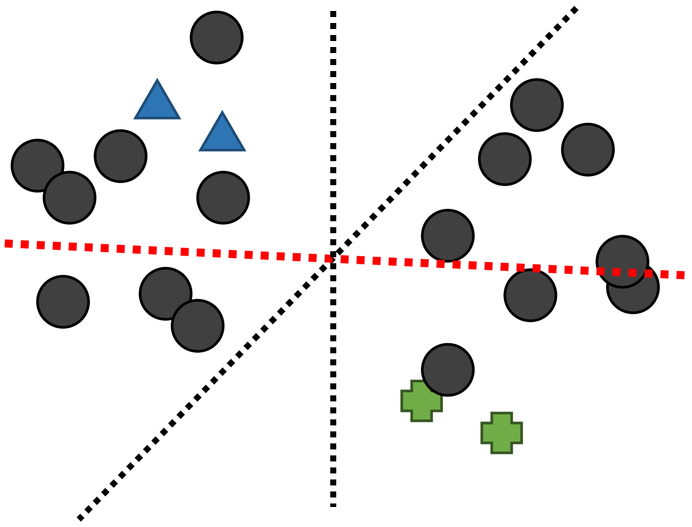
Assumption:
Instances in the same cluster are more likely to have the same label



Looking at all of the data, we might better evaluate the quality of different separating hyperplanes

Types of Learning

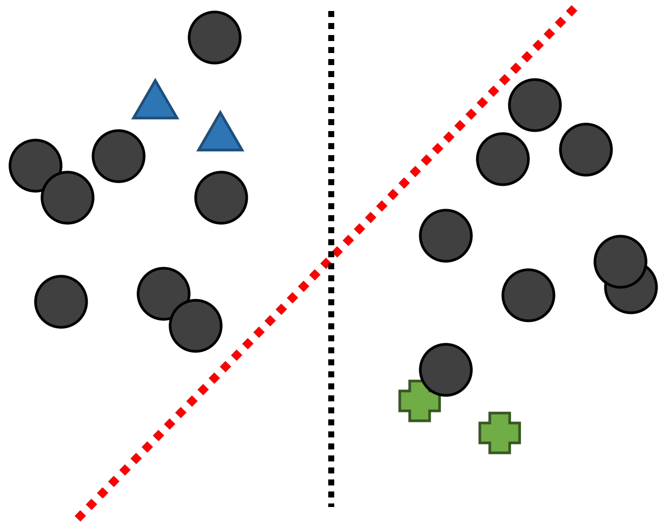
Assumption:
Instances in the same cluster are more likely to have the same label



A line that cuts through both clusters is probably not a good separator

Types of Learning

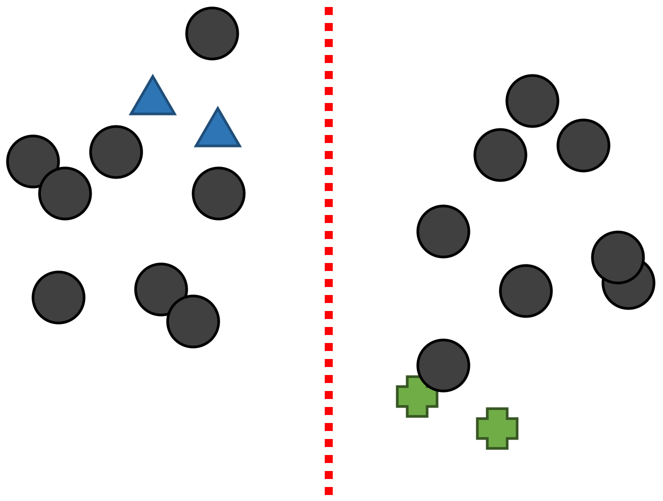
Assumption:
Instances in the same cluster are more likely to have the same label



A line with a small margin between clusters probably has a small margin on labeled data

Types of Learning

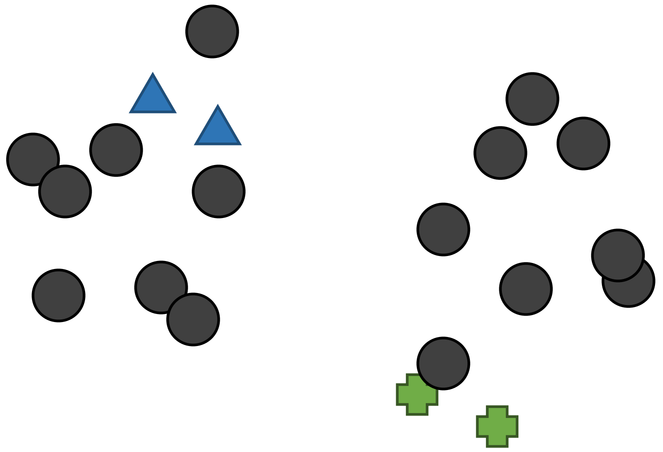
Assumption:
Instances in the same cluster are more likely to have the same label



This would be a pretty good separator, if our assumption is true

Types of Learning

Assumption:
Instances in the same cluster are more likely to have the same label



Our assumption might be wrong:
But with no other information, incorporating unlabeled data probably better than ignoring it!

Semi-Supervised Learning

Semi-supervised learning requires some assumptions about the distribution of data and its relation to labels

Common assumption:

Instances are more likely to have the same label if they are similar (e.g., have a small distance)

Semi-Supervised Learning

Semi-supervised learning is a good idea if your labeled dataset is small, and you have a large amount of unlabeled data

If your labeled data is large, then semi-supervised learning less likely to help…

- How large is “large”? Use learning curves to determine if you have enough data.
- It’s possible for semi-supervised methods to hurt! Be sure to evaluate.

Semi-Supervised Learning

Terminology: both the labeled and unlabeled data that you use to build the classifier are still considered training data

- Though you should distinguish between labeled/ unlabeled

Test data and validation data are labeled

- As always, don’t include test/validation data in training

Label Propagation

Label propagation is a semi-supervised algorithm similar to K-nearest neighbors

Each instance has a probability distribution over class labels: $P(Y_i)$ for instance i

- Labeled instances: $P(Y_i=y) = 1$ if the label is y
  $= 0$ otherwise
- Unlabeled instances: $P(Y_i=y) = 1/S$ initially,
  where S is the
  number of classes

Label Propagation

Algorithm iteratively updates $P(Y_i)$ for unlabeled instances

$P(Y_i=y) = \frac{1}{K} \sum_{j \in N(i)} P(Y_j=y)$

where N(i) is the set of K-nearest neighbors of i

- i.e., an average of the labels of the neighbors

One iteration of the algorithm performs an update of $P(Y_i)$ for every instance

- Stop iterating once $P(Y_i)$ stops changing

Label Propagation

Lots of variants of this algorithm

Commonly, instead of a simple average of the nearest neighbors, a weighted average is used, where neighbors are weighted by their distance to the instance

- In this version, need to be careful to renormalize values after updates so $P(Y_i)$ still forms a distribution that sums to 1

Label Propagation

Label propagation is often used as an initial step for assigning labels to all the data

- You would then still train a classifier on the data to make predictions of new data
- For training the classifier, you might only include instances where $P(Y_i)$ is sufficiently high

Self-Training

Self-training is the oldest and perhaps simplest form of semi-supervised learning

General idea:

1. Train a classifier on the labeled data, as you normally would
2. Apply the classifier to the unlabeled data
3. Treat the classifier predictions as labels, then re-train with the new data

Self-Training

Usually you won’t include the entire dataset as labeled data in the next step

- High risk of included mislabeled data

Instead, only include instances that your classifier predicted with high confidence

- e.g., high probability or high score
- Similar to thresholding to get high precision

This process can be repeated until there are no new instances with high confidence to add

Self-Training

In generative models, an algorithm closely related to self-training is commonly used, called **expectation maximization (EM).**

- We’ll start with Naïve Bayes as an example of a generative model to demonstrate EM

Naïve Bayes

Learning probabilities in Naïve Bayes:

$$P(X_j = x \mid Y = y) = \frac{\text{\# instances with label } y \text{ where feature } j \text{ has value } x}{\text{\# instances with label } y}$$

Naïve Bayes

Learning probabilities in Naïve Bayes:

$$P(X_j{=}x \mid Y{=}y) = \frac{\sum_{i=1}^{N} \mathrm{I}(Y_i{=}y)\ \mathrm{I}(X_{ij}{=}x)}{\sum_{i=1}^{N} \mathrm{I}(Y_i{=}y)}$$

where $\mathrm{I}()$ is an *indicator* function that outputs 1 if the argument is true and 0 otherwise

Naïve Bayes

Learning probabilities in Naïve Bayes:

$$P(X_j=x \mid Y=y) = \frac{\sum_{i=1}^{N} I(Y_i=y)\, I(X_{ij}=x)}{\sum_{i=1}^{N} I(Y_i=y)}$$

where $I()$ is an *indicator* function that outputs 1 if the argument is true and 0 otherwise

Naïve Bayes

Learning probabilities in Naïve Bayes:

$$P(X_j=x \mid Y=y) = \frac{\sum_{i=1}^{N} P(Y_i=y)\, I(X_{ij}=x)}{\sum_{i=1}^{N} P(Y_i=y)}$$

We can also estimate this for unlabeled instances!

$P(Y_i=y)$ is the probability that instance i has label y

- For labeled data, this will be the same as the indicator function (1 if the label is actually y, 0 otherwise)

Naïve Bayes

Estimating $P(Y_i=y)$ for unlabeled instances?

Estimate $P(Y=y \mid X_i)$

- Probability of label y given feature vector Xi

Bayes' rule:

$$P(Y=y \mid X_i) = \frac{P(X_i \mid Y=y)\, P(Y=y)}{P(X_i)}$$

Naïve Bayes

Estimating $P(Y_i=y)$ for unlabeled instances?

Estimate $P(Y=y \mid X_i)$

- Probability of label y given feature vector Xi

Bayes' rule:

$$P(Y=y \mid X_i) = \frac{P(X_i \mid Y=y)\, P(Y=y)}{P(X_i)}$$

- These are the parameters learned in the training step of Naïve Bayes

Naïve Bayes

Estimating $P(Y_i=y)$ for unlabeled instances?

Estimate $P(Y=y \mid X_i)$

- Probability of label y given feature vector Xi

Bayes' rule:

$$P(Y=y \mid X_i) = \frac{P(X_i \mid Y=y)\, P(Y=y)}{P(X_i)}$$

- Last time we said not to worry about this, but now we need it

Naïve Bayes

Estimating $P(Y_i=y)$ for unlabeled instances?

Estimate $P(Y=y \mid X_i)$

- Probability of label y given feature vector Xi

Bayes' rule:

$$P(Y=y \mid X_i) = \frac{P(X_i \mid Y=y)\, P(Y=y)}{\Sigma_{y'}\, P(X_i \mid Y=y')\, P(Y=y')}$$

- Equivalent to the sum of the numerators of each possible y value
- Called *marginalization* (but not covered here)

Naïve Bayes

Estimating $P(Y_i=y)$ for unlabeled instances?

Estimate $P(Y=y \mid X_i)$

- Probability of label y given feature vector Xi

Bayes' rule:

$$P(Y=y \mid X_i) = \frac{P(X_i \mid Y=y)\, P(Y=y)}{\sum_{y'} P(X_i \mid Y=y')\, P(Y=y')}$$

In other words: calculate the Naïve Bayes prediction value for each class label, then adjust to sum to 1

Semi-Supervised Naïve Bayes

1. Initially train the model on the labeled data
   - Learn P(X | Y) and P(Y) for all features and classes
2. Run the EM algorithm (next slide) to update P(X | Y) and P(Y) based on unlabeled data
3. After EM converges, the final estimates of P(X | Y) and P(Y) can be used to make classifications

Expectation Maximization (EM)

The EM algorithm iteratively alternates between two steps:

1. Expectation step (E-step)

Calculate $P(Y=y \mid X_i) = \dfrac{P(X_i \mid Y=y)\, P(Y=y)}{\underbrace{\Sigma_{y'} P(X_i \mid Y=y')\, P(Y=y')}_{\text{These parameters come from the previous iteration of EM}}}$

for every unlabeled instance

$P(Y=y \mid X_i) = \mathbb{I}(Y_i=y)$ for labeled instances

Expectation Maximization (EM)

The EM algorithm iteratively alternates between two steps:

2. Maximization step (M-step)

Update the probabilities P(X | Y) and P(Y), replacing the observed counts with the **expected values** of the counts

- Equivalent to $\sum_i P(Y=y \mid X_i)$

Expectation Maximization (EM)

The EM algorithm iteratively alternates between two steps:

2. Maximization step (M-step)

$$P(X_j{=}x \mid Y{=}y) = \frac{\sum_i P(Y{=}y \mid X_i)\, \mathbb{I}(X_{ij}{=}x)}{\sum_i \underbrace{P(Y{=}y \mid X_i)}_{\text{These values come from the E-step}}}$$

for each feature j
and each class y

Expectation Maximization (EM)

The EM algorithm iteratively alternates between two steps:

2. Maximization step (M-step)

$$P(Y=y) = \frac{\sum_i P(Y=y \mid X_i)}{N}$$

(the # of instances)

for each class y

Expectation Maximization (EM)

The EM algorithm iteratively alternates between two steps:

2. Maximization step (M-step)

Why is it called *maximization*?

- The updates are maximizing the likelihood of the variables
- Same idea as the logistic regression objective function

Expectation Maximization (EM)

An iteration of the EM algorithm corresponds to both an E-step followed by an M-step

- Each E-step uses the parameters learned from the previous M-step
- Each M-step uses the expected values learned from the previous E-step

The algorithm converges when the E-step and M-step are identical to the previous iteration

- The EM algorithm will always converge

Semi-Supervised Naïve Bayes

1. Initially train the model on the labeled data
    - Learn P(X | Y) and P(Y) for all features and classes
2. Run the EM algorithm to update P(X | Y) and P(Y) based on unlabeled data
3. After EM converges, the final estimates of P(X | Y) and P(Y) can be used to make classifications

Semi-Supervised Naïve Bayes

A potential challenge if the size of unlabeled data is much larger than labeled data:

The M-step (updating the probabilities) will be mostly influenced by the unlabeled data

- The labeled data might not have much effect

Modification to EM for semi-supervised NB:

- Start with a small amount of unlabeled data
- Gradually increase the amount of unlabeled data in later iterations of EM

Expectation Maximization (EM)

In general, EM can be used to optimize parameters of any generative model with **latent variables** (variables with unknown value)

- The Y labels of the unlabeled data are the latent variables in semi-supervised Naïve Bayes

Expectation Maximization

A variant of EM:

In the M-step, replace the expected value with 1 if it is the most probable class and 0 otherwise

- This ends up being identical to self-training

Sometimes called “hard” EM, while the traditional version is called “soft” EM

Expectation Maximization

EM can be used for *any* latent variables

- Doesn't matter if some are labeled and others are unlabeled
- EM can work even if the data is entirely unlabeled!

Generative models are often used for unsupervised learning / clustering

- EM is the unsupervised learning algorithm

Unsupervised Naïve Bayes

1. Need to set the number of latent classes
2. Initially define the parameters *randomly*
   - Randomly initialize P(X | Y) and P(Y) for all features and classes
3. Run the EM algorithm to update P(X | Y) and P(Y) based on unlabeled data
4. After EM converges, the final estimates of P(X | Y) and P(Y) can be used for clustering