COMS W4701: Artificial Intelligence

Lecture 9b: Probabilistic Learning

Tony Dear, Ph.D.
Department of Computer Science
School of Engineering and Applied Sciences

Today

- Machine learning foundations
- Bayes net learning
- Laplace smoothing
- Expectation-maximization

Learning

- Given data generated by an unknown function f, a learning agent seeks to find a **hypothesis** $\hat{f}$ to "best" approximate f

- f may be a function of input variables / predictors / features
- Alternatively, f may be a joint probability distribution

- $\hat{f}$ can be used for *prediction* (e.g., classification, regression) of data inputs
- $\hat{f}$ can be used for *inference* of relationships between inputs and outputs
- $\hat{f}$ can be used to find *patterns* in input data (e.g., clustering)

Methods for Learning

- **Parametric methods** first specify a class of functions (e.g., linear combination of features) and then estimate their *parameters*
- In Bayes nets, maximum likelihood can be used for **density estimation**

- **Nonparametric methods** do not make assumptions about the model class, can potentially fit many more functions (e.g., k-nearest neighbors)
- May be more flexible, but may be harder to learn or less interpretable

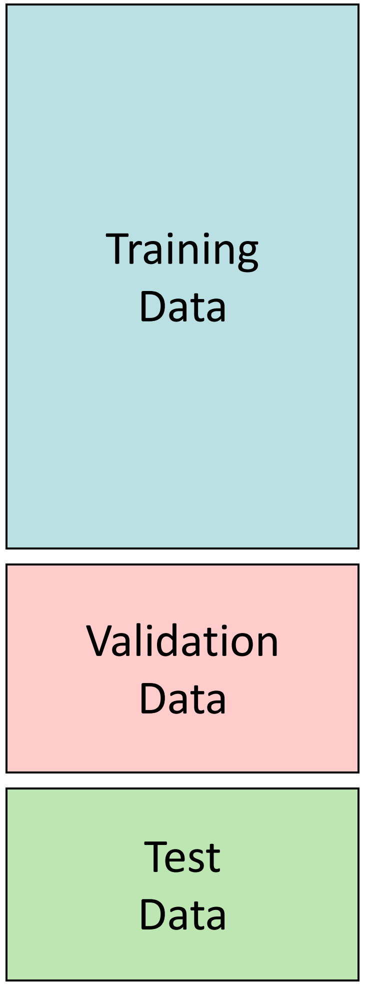
- Learning may be supervised, unsupervised, or somewhere in between depending on whether output values are available

Performance Evaluation

- We first specify hypothesis space in which we search/optimize
- We then train a model on using a given set of *training data*

- Next, find ways to slightly modify model to generalize it
- Methods: Cross-validation, hand-tuning, grid search, etc.
- Use *validation data to* tune any **hyperparameters**

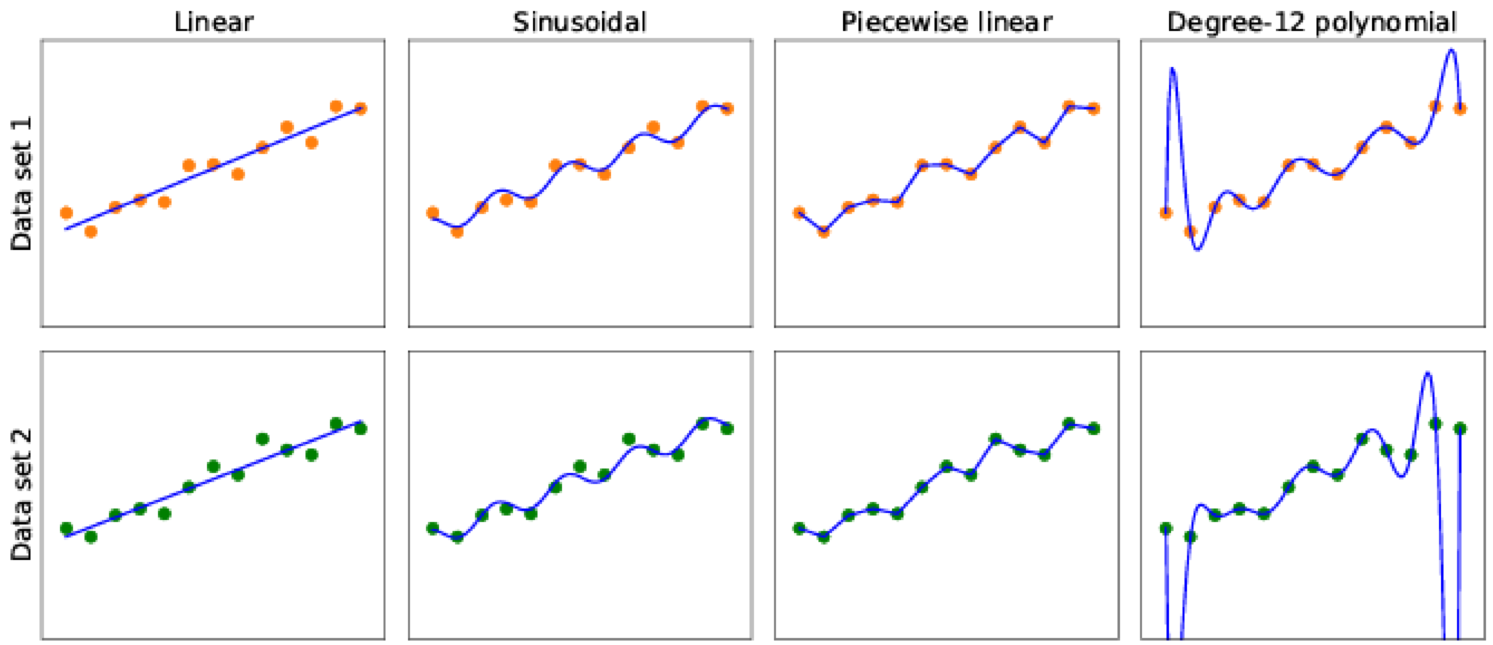
- Tuning lowers training accuracy, increases validation accuracy
- Final model evaluation is done on a new set of *test data*



Bias-Variance Tradeoff

- More complex models will generally better fit the given data, resulting in higher accuracies, lower error, less *underfitting*, and lower **bias**
- Tradeoff: Models that are too flexible may *overfit* to noisy or unrepresentative training data, resulting in greater **variance** in the learned models given different data sets

Higher bias,
lower variance



Lower bias,
higher variance

Other Learning Considerations

- Models with higher complexity can be used for greater variety of problems
- E.g., deep neural networks are very successful at hard vision/NLP tasks

- Tradeoff: More data required, more computation for learning a good model
- Higher complexity also generally leads to lower interpretability
- May be good for prediction, but not so good for inference and explainability

- When data is high-dimensional, data sets become very sparse, leading to high variance
- Feature selection and dimensionality reduction techniques can help
- Learning methods must also be able to handle imperfect, noisy, and missing data

Bayes Net Learning

- Suppose we want to learn the *parameters* θ (probabilities) of a Bayes net
- Hypothesis space is the set of all possible CPTs / joint distributions

- **Maximum-likelihood** of independent and identically distributed data $\boldsymbol{d} = (d_1, \dots, d_N)$:

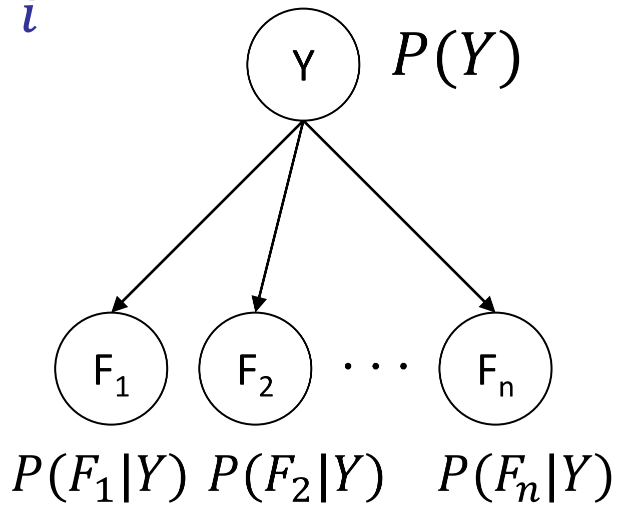
$$\max_{\theta} \Pr(\boldsymbol{d}|\theta) = \max \prod_{i=1}^{N} P(d_i|\theta)$$

- $P(d_i|\theta)$ is the Bayes net joint probability of the data point d_i

- We can perform optimization to solve this, but solution is (again) intuitive!
- MLE parameters are just *frequencies*, and each CPT can be learned *independently*

Naïve Bayes Models

- **Naïve Bayes** model: Fork structure with single class or label variable Y
- Each label is correlated with many different features F_i



- Joint probabilities: $P(y, f_1, \dots, f_n) = P(y) \prod_{i=1}^{n} P(f_i|y)$

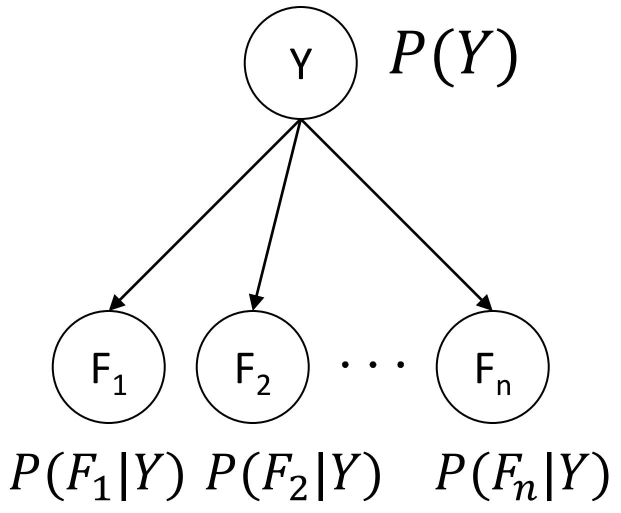
- A common inference task is **classification**
- Suppose are given an input *feature vector* (all f_i)

- Most likely class label: $y = \underset{y}{\operatorname{argmax}} P(y|f_1, \dots, f_n) = \underset{y}{\operatorname{argmax}} P(y) \prod_{i=1}^{n} P(f_i|y)$

Example: Spam Filtering

- $P(Y) =$ Spam prior, or likelihood that generic email is spam
- $P(F_i|Y) =$ Conditional probabilities that a word F_i appears or doesn't appear in a spam or ham email

- **"Bag-of-words"** model: Each word is treated independently, no dependency on other or nearby words



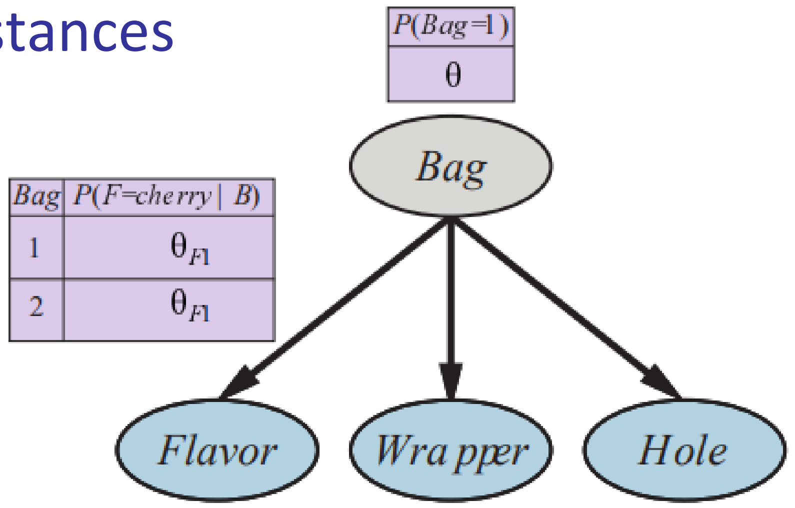
	$\boldsymbol{P(Y)}$
Spam	1/3
Ham	2/3

Word	**Hey**	**would**	**you**	**like**	**to**	**lose**	**weight**
$\boldsymbol{P(f|\text{spam})}$	0.00002	0.00069	0.00881	0.00086	0.01517	0.00008	0.00016
$\boldsymbol{P(f|\text{ham})}$	0.00021	0.00084	0.00304	0.00083	0.01339	0.00002	0.00002

- Prediction: $$\underset{spam,ham}{\operatorname{argmax}}\left\{P(spam)\prod_{i=1}^{n}P(f_i|spam)\,,P(ham)\prod_{i=1}^{n}P(f_i|ham)\right\} = spam$$

Learning Naïve Bayes Models

- Example: Candy bag prediction given candy features
- Suppose data $\boldsymbol{d}$ has b_1 bag 1 instances and $n - b_1$ bag 2 instances



- Data likelihood for *Bag* class: $P(\boldsymbol{d}|\theta) = \theta^{b_1}(1-\theta)^{n-b_1}$
- It is often convenient to work with *log likelihood function*:

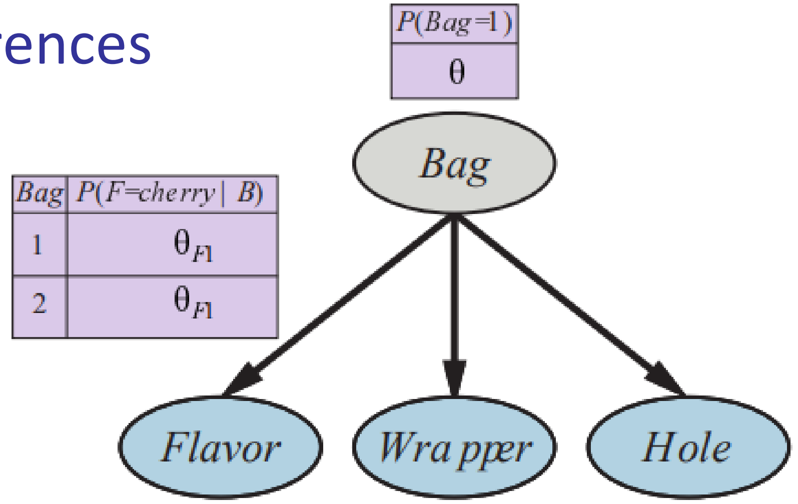
$$L(\boldsymbol{d}|\theta) = \log P(\boldsymbol{d}|\theta) = \log \theta^{b_1} + \log(1-\theta)^{n-b_1}$$

$$= b_1 \log \theta + (n - b_1)\log(1-\theta)$$

- To maximize wrt θ, solve $\frac{dL}{d\theta} = 0$: $\quad \frac{dL}{d\theta} = \frac{b_1}{\theta} - \frac{n-b_1}{1-\theta} = 0 \quad \Rightarrow \quad \theta = \frac{b_1}{n}$

Learning Naïve Bayes Models

- Can write likelihoods for each feature and optimize similarly
- Results always come down to simply counting data occurrences



- Maximum-likelihood parameter estimates:

$$\theta = \frac{N(bag=1)}{N(samples)} \qquad \theta_{F1} = \frac{N(bag=1, flavor=cherry)}{N(bag=1)}$$

$$1-\theta = \frac{N(bag=2)}{N(samples)} \qquad \theta_{F2} = \frac{N(bag=2, flavor=cherry)}{N(bag=2)}$$

- Suppose training data contains no instance of $(Bag = 1, F = cherry)$ candies
- When predicting on future instances, the classifier will give $P(Bag = 1) = 0$ for any input containing $F = cherry$, regardless of other features!

Laplace Smoothing

- **Data fragmentation** in Bayes nets: Each parameter is estimated using only a subset of the training data, leading to *overfitted* estimates
- Problem worsens as the CPTs become larger (higher dimensionality)

- **Laplace smoothing**: Reduce overfitting by adding a “virtual count” α

$$\widehat{P(X)} = \frac{n(x) + \alpha}{N + \alpha|X|} \xrightarrow{\alpha \to 0} \widehat{P(X)} = \frac{n(x)}{N}$$

Original maximum likelihood estimator (average)

$$\widehat{P(X)} = \frac{n(x) + \alpha}{N + \alpha|X|} \xrightarrow{\alpha \to \infty} \widehat{P(X)} = \frac{1}{|X|}$$

Uniform prior over all possibilities of X (ignore all samples)

Example: Laplace Smoothing

- Suppose we have the following training data set:

F_1	F_2	F_3	Y
$+f_1$	$+f_2$	$-f_3$	$+y$
$+f_1$	$-f_2$	$-f_3$	$+y$
$-f_1$	$+f_2$	$-f_3$	$+y$
$+f_1$	$+f_2$	$+f_3$	$-y$
$-f_1$	$+f_2$	$+f_3$	$-y$
$-f_1$	$+f_2$	$-f_3$	$-y$

- Estimates of class priors: $P(+y) = P(-y) = 0.5$
- MLE would give $P(+f_3|+y) = 0$ and $P(-f_2|-y) = 0$

- Suppose we use $\alpha = 1$ smoothing instead:

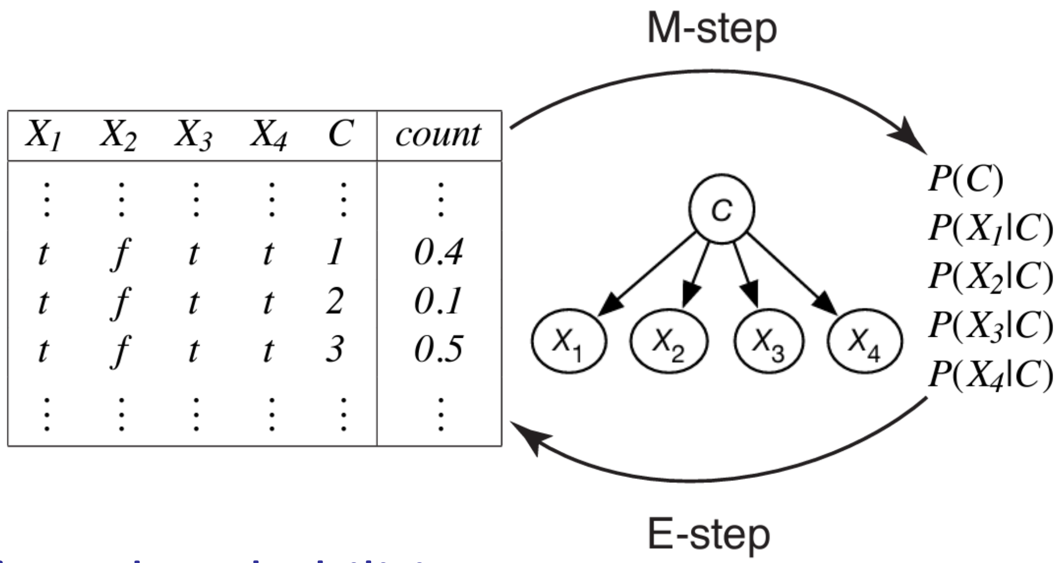
$P(+f_1\|+y) = \frac{2+1}{3+2} = \frac{3}{5}$	$P(+f_2\|+y) = \frac{2+1}{3+2} = \frac{3}{5}$	$P(+f_3\|+y) = \frac{0+1}{3+2} = \frac{1}{5}$
$P(+f_1\|-y) = \frac{1+1}{3+2} = \frac{2}{5}$	$P(+f_2\|-y) = \frac{3+1}{3+2} = \frac{4}{5}$	$P(+f_3\|-y) = \frac{2+1}{3+2} = \frac{3}{5}$

$$\widehat{P(X)} = \frac{n(x) + \alpha}{N + \alpha|X|}$$

Expectation-Maximization

- As with HMM learning, our data may be incomplete due to *hidden* or *latent* variables
- But still important for reducing the number of parameters and data needed to learn

- Recall the *expectation-maximization* approach:
- **Expectation**: Use *inference* with current parameters to compute *expected* counts
- **Maximization**: Use expected (and observed) counts to compute new parameters



X_1	X_2	X_3	X_4	C	*count*
⋮	⋮	⋮	⋮	⋮	⋮
t	*f*	*t*	*t*	*1*	*0.4*
t	*f*	*t*	*t*	*2*	*0.1*
t	*f*	*t*	*t*	*3*	*0.5*
⋮	⋮	⋮	⋮	⋮	⋮

- Can initialize with some prior or guess of the inferred probabilities
- EM will converge to a local maximum of the data likelihood

Example: Expectation-Maximization

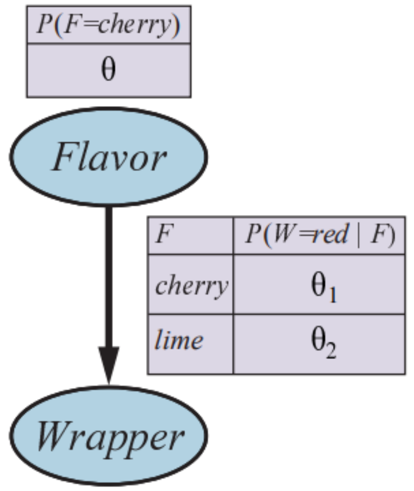
- Suppose we currently have $P(F) = (0.6, 0.4), P(W = red|F) = (0.6, 0.4)$, and $P(W = green|F) = (0.4, 0.6)$

- Suppose *Flavor* is hidden in the data set, but *Wrapper* is accessible
- Inference computation of *Flavor* conditioned on *Wrapper values*:

$$P(F|red) \propto P(F)P(red|F) = \begin{pmatrix}0.6\\0.4\end{pmatrix} \times \begin{pmatrix}0.6\\0.4\end{pmatrix} \propto \begin{pmatrix}0.69\\0.31\end{pmatrix}$$

$$P(F|green) \propto P(F)P(green|F) = \begin{pmatrix}0.6\\0.4\end{pmatrix} \times \begin{pmatrix}0.4\\0.6\end{pmatrix} \propto \begin{pmatrix}0.5\\0.5\end{pmatrix}$$

- These are the *estimated frequencies* of the different *Flavor* values in our data set!



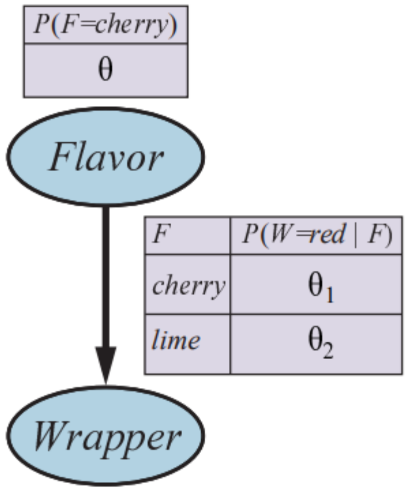
Example: Expectation-Maximization

- Suppose our data set contains 545 red and 455 green wrapper instances

$$P(F|red) = \begin{pmatrix}0.69\\0.31\end{pmatrix} \quad P(F|green) = \begin{pmatrix}0.5\\0.5\end{pmatrix}$$

- “Augmented” data set (E-step):

Flavor	***Wrapper***	***N***
cherry	*red*	376.05
cherry	*green*	227.5
lime	*red*	168.95
lime	*green*	227.5



- New parameters (M-step):

$$P(F) = \frac{1}{1000}\begin{pmatrix}376.05 + 227.5\\168.95 + 227.5\end{pmatrix} = \begin{pmatrix}0.604\\0.396\end{pmatrix}$$

$$P(W|F = cherry) = \frac{1}{603.55}\begin{pmatrix}376.05\\227.5\end{pmatrix} = \begin{pmatrix}0.623\\0.377\end{pmatrix} \quad P(W|F = lime) = \frac{1}{396.45}\begin{pmatrix}168.95\\227.5\end{pmatrix} = \begin{pmatrix}0.426\\0.574\end{pmatrix}$$

EM Considerations

- Inference can be done analytically or approximately (sampling)
- We can even *interleave* MCMC and EM iterations by treating generated samples as complete observation, giving us *approximate* expectations

- As with Baum-Welch, data likelihood is guaranteed to increase in each iteration, converging at a local maximum

- However, improvements tend to slow down after a few iterations
- Can combine with local search or gradient-based methods

Summary

- Statistical learning involves learning a hypothesis to approximate a hidden function
- Can be used for prediction, inference, pattern detection, and other tasks

- Many tradeoffs involving bias and variance, flexibility and generalizability, expressiveness and interpretability, and complexity of learning
- Regularization can help generalize models at the expense of training accuracy

- MLE is a standard technique for learning Bayes nets with complete data
- Smoothing can help regularize naïve Bayes model learning
- EM can be used when data contains hidden variables