

CS534: Machine Learning

Spring 2009

Course Information

Instructor: Xiaoli Fern

Office hours:

MWF 10-11 or by appointment

Grader: Paul Wilkins pwilkins05@gmail.com

Class Web Page:

web.engr.oregonstate.edu/~xfern/classes/cs534

Class email list:

cs534-sp09@engr.orst.edu

Course Overview

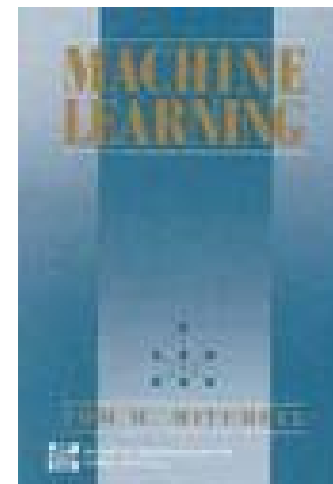
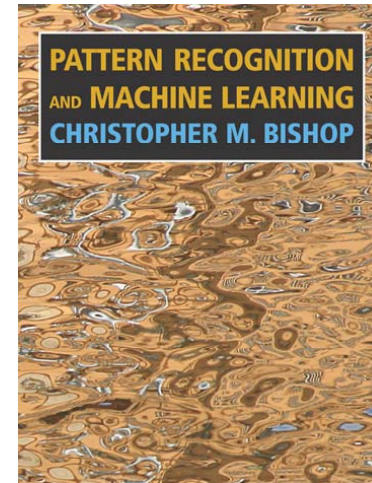
- Covers a wide range of machine learning techniques
 - Perceptron, Logistic regression, LDA, Decision tree, Naïve Bayes, KNN, SVM, Hidden Markov Models, EM, K-means, Mixture of Gaussian, PCA, etc...
- You will learn about algorithms, related theories and applications

Prerequisites

- Knowledge of basic CS concepts such as data structure, search strategies, computational complexity ...
- Familiarity with basic probability theory – this is very important
 - If you are not comfortable with this, please read the related review sections in the text book
 - Andrew Moore also has a collection of tutorials that are very useful and very accessible
- Some basic statistics
- Calculus and linear algebra
 - Matrix cookbook is a good resource for linear algebra

Text

- Required: Pattern Recognition and Machine Learning by Bishop
<http://research.microsoft.com/en-us/um/people/cmbishop/prml/>
- Optional reference: *Machine Learning* by Tom Mitchell



Grading

- 30% written and implementation assignments
- 25% final project
 - Start early
 - Best combined with your own research
 - Aim for conference publication quality (June 5th Deadline for NIPS, June 26th Deadline for ICDM09)
- Midterm – 20%
- Final exam – 25%

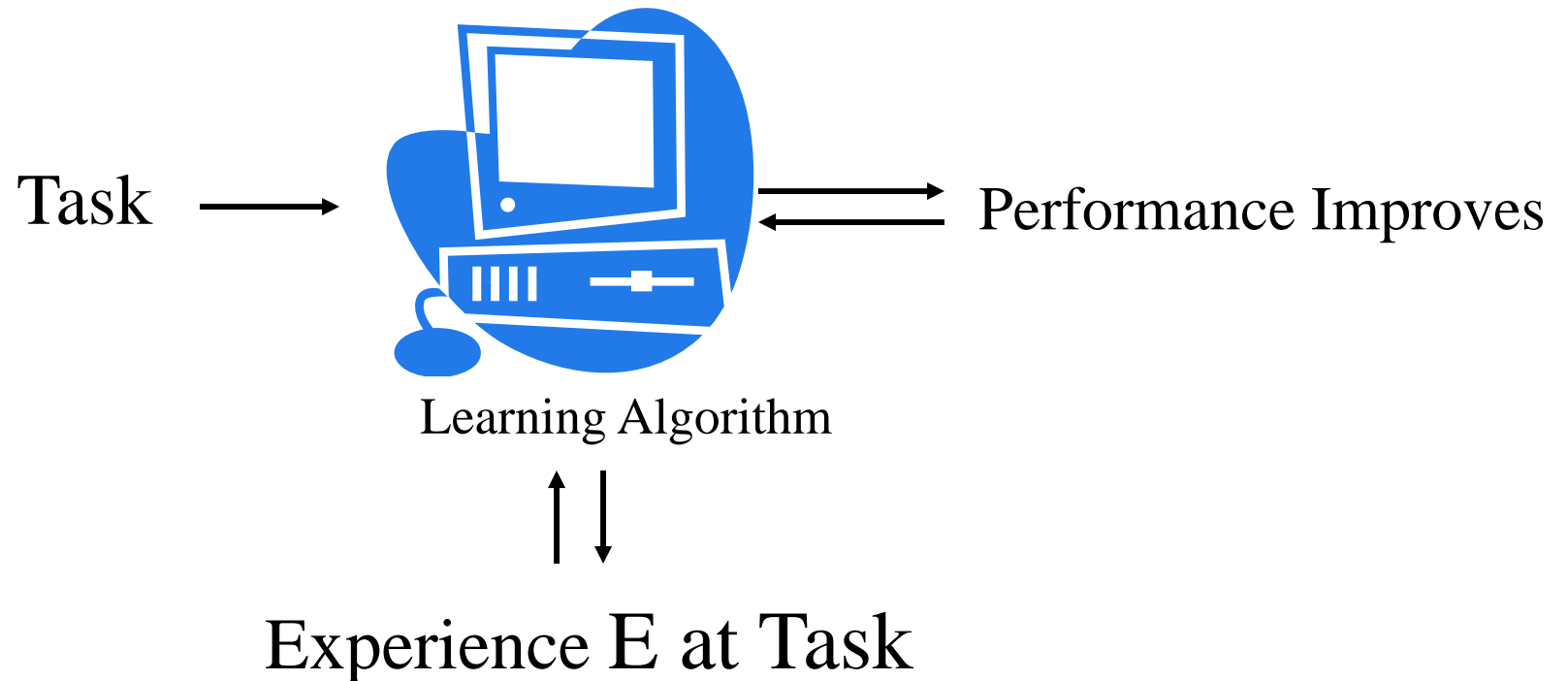
Homeworks

- Homeworks are due at the beginning of the class
- Late submission will be accepted but discounted by 20% each day for maximum of two days
- Collaborations
 - Discussion allowed
 - Each student has to write their own answers
 - On your homework please list anyone you collaborated with

Assigned Reading

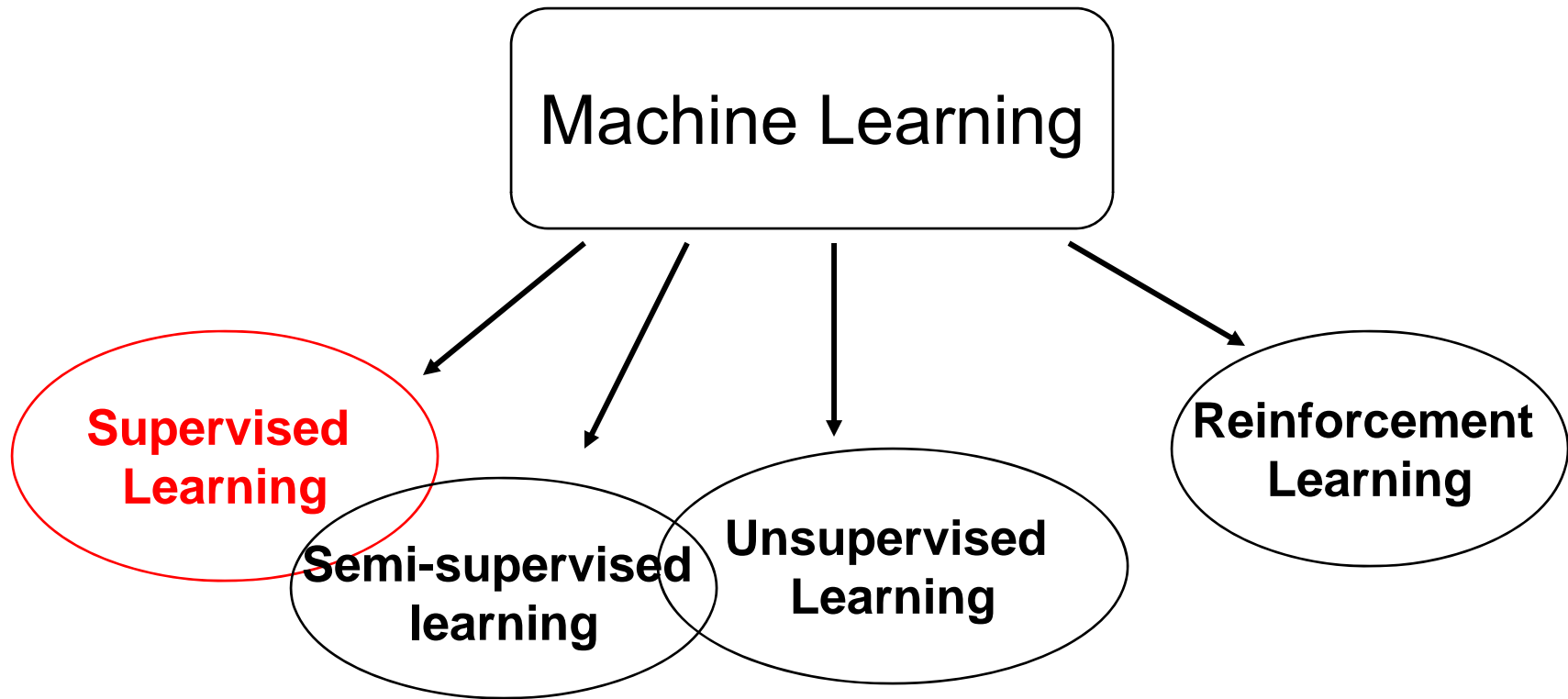
- Chapter 1
 - This chapter may seem a bit random – it provides a number of background concepts that serve as basis for later chapters of the book

What is Machine Learning



Goal: make computers learn from experience

Fields of Study



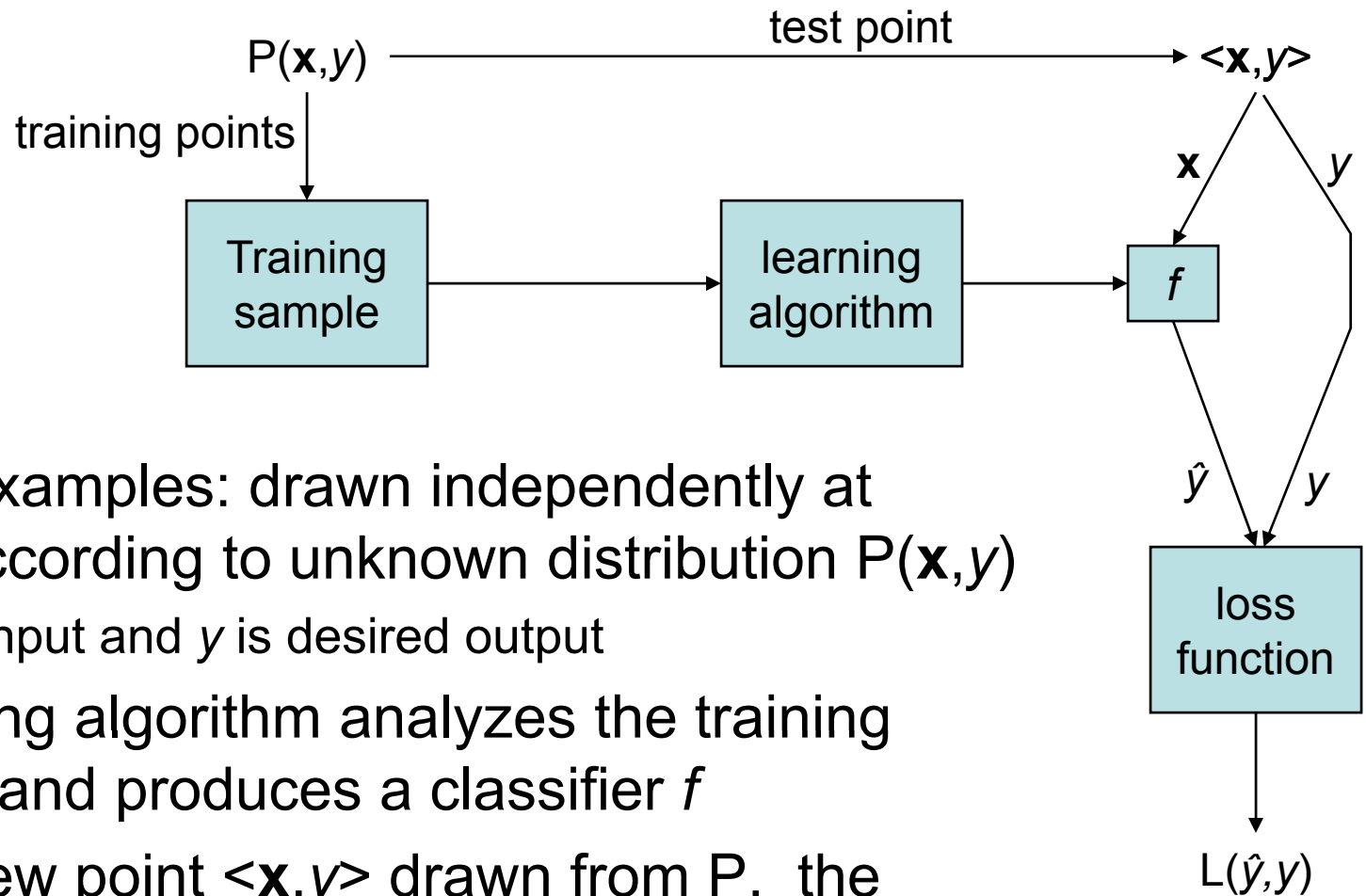
Supervised Learning

- **Given:** training examples $\langle \mathbf{x}, f(\mathbf{x}) \rangle$ for some unknown function f
 - x is the input and $f(x)$ is desired output
- **Goal:** find a good approximation to f
- Example Applications
 - Disease Diagnosis
 - x : properties of patient (symptoms, lab tests)
 - $f(x)$: disease (or maybe, recommended therapy)
 - Face Recognition
 - x : bitmap picture of person's face
 - $f(x)$: name of person
 - Spam Detection
 - x : email message
 - $f(x)$: spam or not spam

Appropriate Applications for Supervised Learning

- Situations where there is no human expert
 - x : bond graph of a new molecule
 - $f(x)$: predicted binding strength to AIDS protease molecule
- Situations where humans can perform the task but can't describe how they do it
 - x : picture of a hand-written character
 - $f(x)$: ascii code of the character
- Situations where the desired function is changing frequently
 - x : description of stock prices and trades for last 10 days
 - $f(x)$: recommended stock transactions
- Situations where each user needs a customized function f
 - x : incoming email message
 - $f(x)$: importance score for presenting to the user (or deleting without presenting)

Formal Setting



- Training examples: drawn independently at random according to unknown distribution $P(\mathbf{x}, y)$
 - \mathbf{x} is the input and y is desired output
- The learning algorithm analyzes the training examples and produces a classifier f
- Given a new point $\langle \mathbf{x}, y \rangle$ drawn from P , the classifier is given \mathbf{x} and predicts $\hat{y} = f(\mathbf{x})$
- The loss $L(\hat{y}, y)$ is then measured
- Goal of the learning algorithm: Find the f that minimizes the *expected loss*

Example: Spam Detection

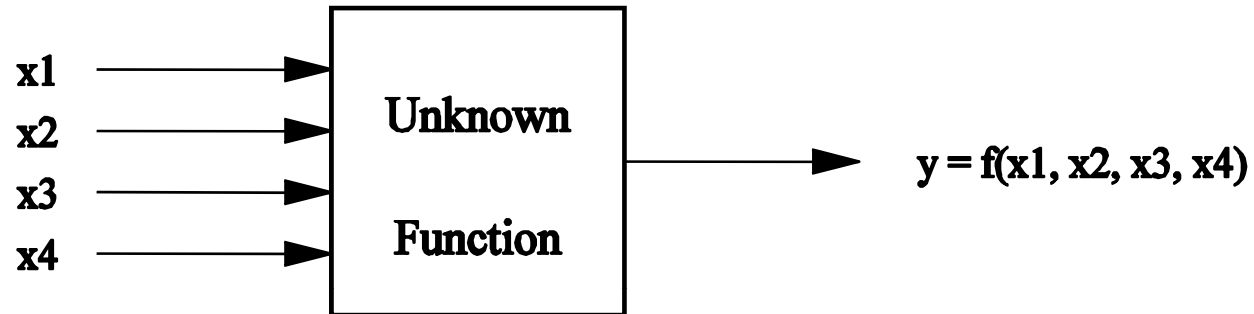
- $P(\mathbf{x}, y)$: distribution of email messages \mathbf{x} and labels y (“spam” or “not spam”)
- Training sample: a set of email messages that have been labeled by the user
- Learning algorithm: what we study in this course!
- f : the classifier output by the learning algorithm
- Test point: A new email message \mathbf{x} (with its true, but hidden, label y)
- loss function $L(\hat{y}, y)$:

predicted label \hat{y}	true label y	
	spam	not spam
spam	0	10
not spam	1	0

Terminology

- **Training example** an example of the form $\langle \mathbf{x}, y \rangle$
 - \mathbf{x} : feature vector
 - y : class label, in $[1, 2, \dots, K]$
- **Training Set** a set of training examples drawn randomly according to $P(\mathbf{x}, y)$
- **Target function** the true mapping from \mathbf{x} to y
- **Hypothesis** a proposed function h believed to be similar to the target function.
- **Test Set** a set of training examples used to evaluate a proposed hypothesis h .
- **Hypothesis space** The space of all hypotheses that can, in principle, be output by a particular learning algorithm

Fundamental Problem of Machine Learning: It is ill-posed



Example	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Learning Appears Impossible

- # of possible Boolean functions over 4 input features: $2^{16} = 65536$.
- Can't figure out which one is correct until we see every possible input-output pair.
- After 7 examples, remaining # of possibilities: 2^9

x_1	x_2	x_3	x_4	y
0	0	0	0	?
0	0	0	1	?
0	0	1	0	0
0	1	0	0	0
1	0	0	0	?
0	0	1	1	1
0	1	0	1	0
1	0	0	1	1
0	1	1	0	0
1	0	1	0	?
1	1	0	0	0
0	1	1	1	?
1	0	1	1	?
1	1	0	1	?
1	1	1	0	?
1	1	1	1	?

Solution: Work with a restricted hypothesis space

- Choose a space of hypotheses H smaller than the space of all possible functions
 - by applying prior knowledge
 - by guessing,
- Examples:
 - simple conjunctive rules
 - *m-of-n* rules
 - linear functions
 - multivariate Gaussian joint probability distributions

Illustration: Simple Conjunctive Rules

- Simple conjunctions (no negation): 16 possible hypotheses
- However, no one explains the data. The same is true for simple clauses

Example	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Rule	Counterexamples
$\text{true} = y$	1,2,5,6,7
$x_1 = y$	3,6
$x_2 = y$	2,3,4,5,6,7
$x_3 = y$	1,4,5
$x_4 = y$	7
$x_1 \wedge x_2 = y$	3,4
$x_1 \wedge x_3 = y$	3,4
$x_1 \wedge x_4 = y$	3
$x_2 \wedge x_3 = y$	4
$x_2 \wedge x_4 = y$	3,7
$x_3 \wedge x_4 = y$	4
$x_1 \wedge x_2 \wedge x_3 = y$	3,4
$x_1 \wedge x_2 \wedge x_4 = y$	3,4
$x_1 \wedge x_3 \wedge x_4 = y$	3,4
$x_2 \wedge x_3 \wedge x_4 = y$	3,4
$x_1 \wedge x_2 \wedge x_3 \wedge x_4 = y$	3,4

A larger hypothesis space

m out of n rules

- At least m of the n variables must be true
- 32 possible rules
- One consistent rule!

m of n	Counterexample			
	1-of	2-of	3-of	4-of
{X ₁ }	3	-	-	-
{X ₂ }	2	-	-	-
{X ₃ }	1	-	-	-
{X ₄ }	7	-	-	-
{X ₁ ; X ₂ }	3	3	-	-
{X ₁ ; X ₃ }	4	3	-	-
{X ₁ ; X ₄ }	6	3	-	-
{X ₂ ; X ₃ }	2	3	-	-
{X ₂ ; X ₄ }	2	3	-	-
{X ₃ ; X ₄ }	4	4	-	-
{X ₁ ; X ₂ ; X ₃ }	1	3	3	-
{X ₁ ; X ₂ ; X ₄ }	2	3	3	-
{X ₁ ; X ₃ ; X ₄ }	1	***	3	-
{X ₂ ; X ₃ ; X ₄ }	1	5	3	-
{X ₁ ; X ₂ ; X ₃ ; X ₄ }	1	5	3	3

Two Views of Learning

- View 1: **Learning is the removal of the remaining uncertainty**
 - Suppose we *a priori* knew that the target function was an *m-of-n* boolean function. Then we could use the training examples to *deduce* which function it is.
- View 2: **Learning requires guessing a good, small hypothesis class**
 - We can start with a very small class and enlarge it until it contains an hypothesis that fits the data

We could be wrong!

- Our prior “knowledge” might be wrong
- Our guess of the hypothesis class could be wrong
 - The smaller the class, the more likely we are wrong

Two Strategies for Machine Learning

- Develop Languages for Expressing Prior Knowledge
 - Rule grammars, stochastic models, Bayesian networks
 - (Corresponds to the Prior Knowledge view)
- Develop Flexible Hypothesis Spaces
 - Nested collections of hypotheses: decision trees, neural networks, SVMs
 - (Corresponds to the Guessing view)
- In either case we must develop algorithms for finding an hypothesis that fits the data

Key Issues in Machine Learning

- What are good hypothesis spaces?
 - which spaces have been useful in practical applications?
- How to select among different hypothesis spaces?
 - Model selection problem
- How can we optimize accuracy on future data points?
 - This is related to the issue of “overfitting”
 - i.e., the model fits to the peculiarities rather than the generalities of the data
- How can we have confidence in the results? (the statistical question)
 - How much training data is required to find an accurate hypotheses with high probability?
- Are some learning problems computationally intractable? (the computational question)
- How can we formulate application problems as machine learning problems? (the engineering question)

Three Main Approaches

- Learn a classifier: a function f , $\hat{y} = f(\mathbf{x})$
- Learn a conditional distribution: a conditional distribution $P(y | \mathbf{x})$
- Learn the joint probability distribution: $P(\mathbf{x}, y)$
- In the next few lectures we will study one example of each method:
 - Learn a classifier: The perceptron algorithm
 - Learn a conditional distribution: Logistic regression
 - Learn the joint distribution: Linear Discriminant Analysis (LDA)
- First lets consider how one can make predictions given that we learn $P(y | \mathbf{x})$ or $P(\mathbf{x}, y)$

Inferring a Classifier f from $P(y | x)$

- Predict the \hat{y} that minimizes the expected loss:

$$\begin{aligned} f(x) &= \arg \min_{\hat{y}} \overset{\text{conditional expectation}}{E}_{y|x} [L(\hat{y}, y)] \\ &= \arg \min_{\hat{y}} \sum_y P(y | x) L(\hat{y}, y) \end{aligned}$$

Example: Making the spam decision

- Suppose our spam detector predicts $P(y='spam'|\mathbf{x}) = 0.6$
What is the optimal \hat{y} ?

– $\hat{y} = 'spam'$:

$$E[L(\hat{y}, y)] = 0 * 0.6 + 10 * 0.4 = 4$$

– $\hat{y} = 'no spam'$:

$$E[L(\hat{y}, y)] = 1 * 0.6 + 0 * 0.4 = 0.6$$

predicted label \hat{y}	true label y	
	spam	not spam
spam	0	10
not spam	1	0
$P(y \mathbf{x})$	0.6	0.4

- Optimal prediction: 'no spam'
 - Note that 'spam' was more likely according to P but was not selected as the output. Why?

Inferring a classifier from $P(x,y)$

- We can compute the $P(y | x)$ based on $P(x, y)$

$$P(y = k | x) = \frac{P(x, y = k)}{P(x)} = \frac{P(x, y = k)}{\sum_j P(x, y = j)}$$

- Compute \hat{y} using the method from the previous slide