# Artificial Intelligence

CS 165A

Apr 7, 2022

Instructor: Prof. Yu-Xiang Wang

Today

→ Continuous optimization

→ Wrapping up ML

→ Starting PGM

# Recap: Last lecture

- Data splitting
  - Holdout and Cross validation for tuning hyperparameters
- The problem of distribution shift
- Learning a classifier:
  - From 0–1 loss to surrogate losses
  - Logistic loss as an example
- Continuous Optimization with Gradient Descent

# Recap: How do we optimize a continuously differentiable function in general?

- The problem: $\min_{\theta} f(\theta)$

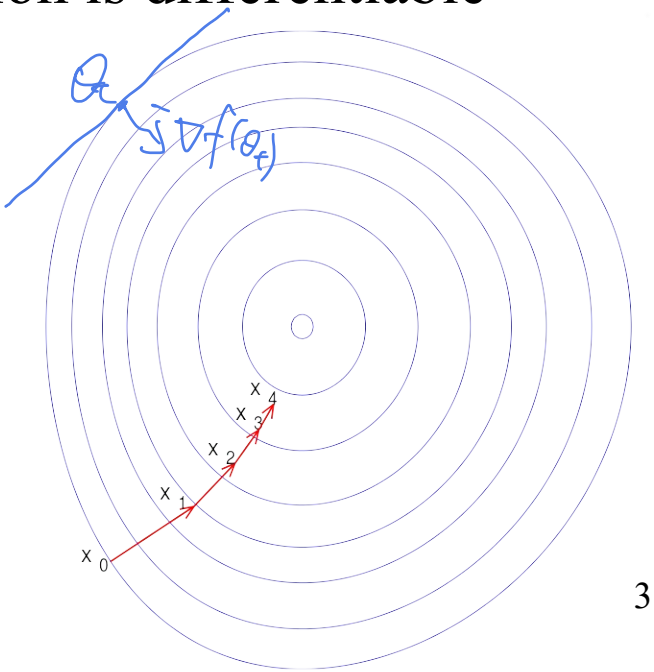
- Let's just optimize it anyway!
  - With gradient descent.

- Assumption: The objective function is differentiable almost everywhere.

learning rate

$$\theta_{t+1} = \theta_t - \eta_t \nabla f(\theta_t)$$

gradient



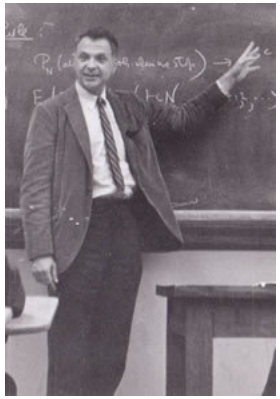
# Recap: Gradient of logistic loss for learning a linear classifier

$$\nabla f(w) = \frac{1}{n} \sum_{i=1}^{n} \frac{\exp(-y_i \cdot x_i^T w)}{1 + \exp(-y_i \cdot x_i^T w)} (-y_i x_i)$$

- Question: What is the time complexity of computing this gradient?

**Ans: O(nd)**

# Recap: Stochastic Gradient Descent (Robbins-Monro 1951)



Herbert Robbins
1915 - 2001

- Gradient descent

$$\theta_{t+1} = \theta_t - \eta_t \nabla f(\theta_t)$$

- Stochastic gradient descent

$$\theta_{t+1} = \theta_t - \eta_t \hat{\nabla} f(\theta_t)$$

- Using a stochastic approximation of the gradient:

$$\mathbb{E}[\hat{\nabla} f(\theta_t) | \theta_t] = \nabla f(\theta_t)$$

$$\mathbf{Var}[\hat{\nabla} f(\theta_t) | \theta_t] \leq \sigma^2$$

The time complexity of each iteration in SGD can be drastically lower!

# Plan for today

- Stochastic Gradient Descent
- Making sense of the SGD updates
- Multiclass classification
- Generative vs Discriminative Models
- Probability notations

# One natural stochastic gradient to consider in machine learning

- Recall that

$$\min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \ell(\theta, (x_i, y_i)) =: L(\theta)$$

- Pick a **single** data point i uniformly at random

  - Use $\nabla_\theta \ell(\theta, (x_i, y_i))$

    $i$ is uniform $\{1, 2, \ldots, n\}$

  - Show that this is an unbiased estimator!

$$\mathbb{E}\left[\nabla_\theta \ell_i(\theta)\right] = \sum_{i=1}^{n} \underbrace{P(i)}_{\frac{1}{n}} \cdot \nabla_\theta \ell_i(\theta) = \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \ell_i(\theta) = \text{full gradient } \nabla L(\theta)$$

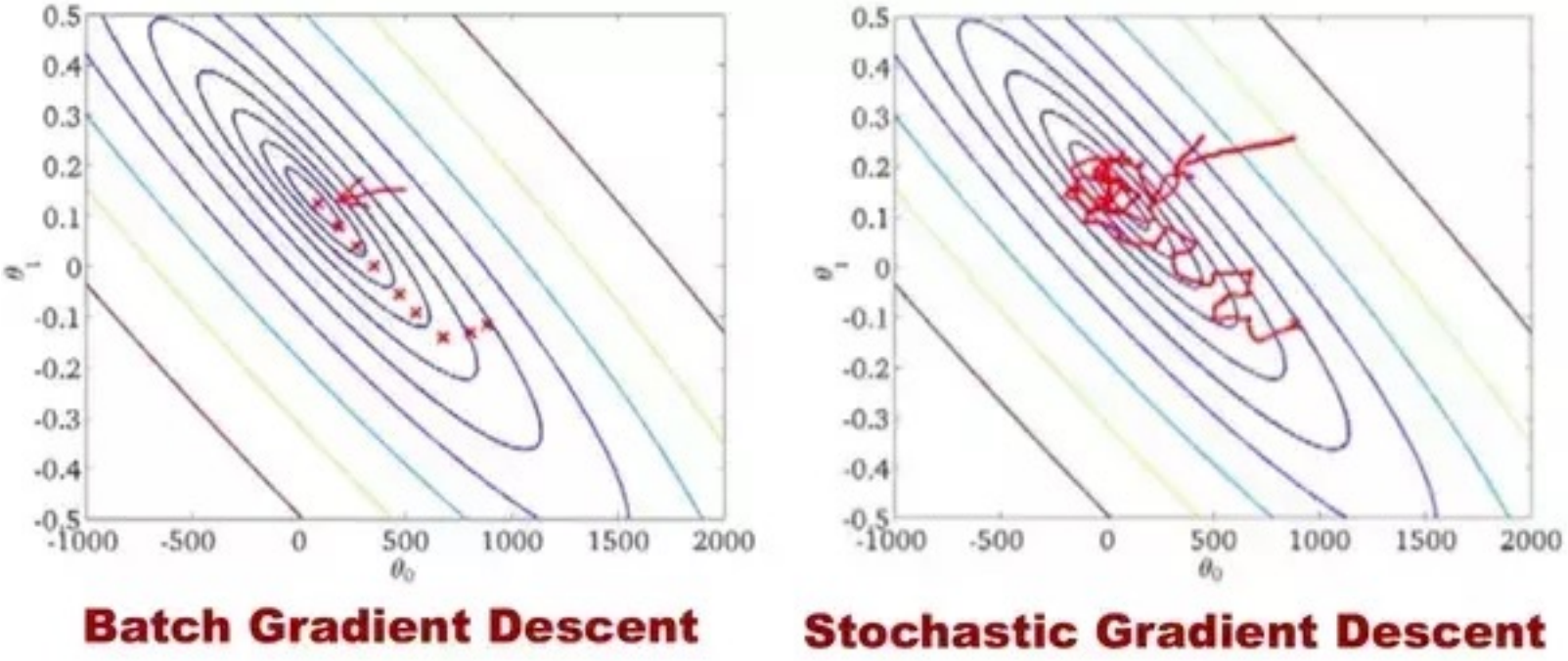
$\nabla \ell_i(w) = s(w^T x_i y_i) \cdot \vec{x}$

Complexity: $O(d)$

$\Downarrow$

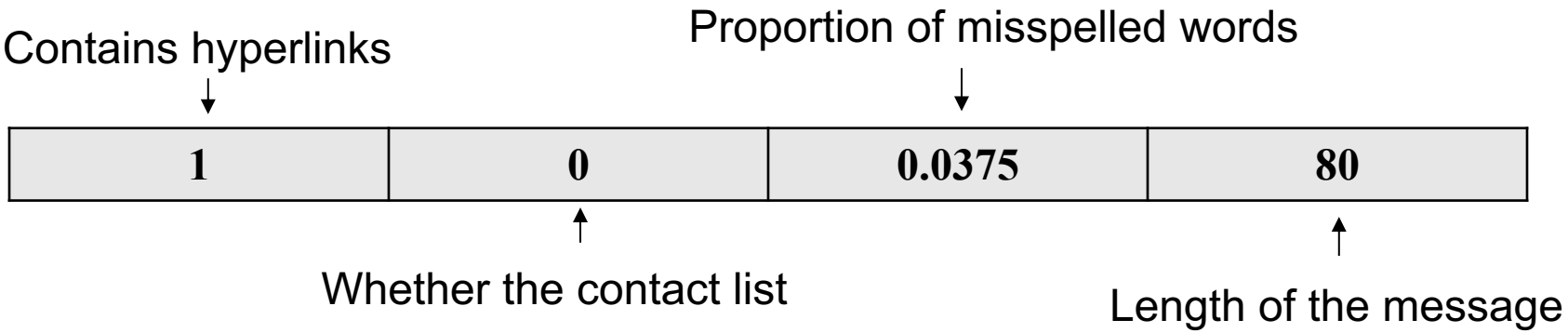
$O(\text{nnz}(x))$

# Illustration of GD vs SGD




**Observation:** With the time gradient descent taking one step. SGD would have already moved many steps.

# Intuition of the SGD algorithm on the "Spam Filter" example




- Score(x) = $w_0 + w_1 * 1(\text{hyperlinks}) + w_2 * 1(\text{contact list}) + w_3 * \text{misspelling} + w_4 * \text{length}$ $= w^T x$ $w \in \mathbb{R}^5$

- Meaning of these weight?
  - The more positive, the more we think the feature is associated with Spam email.
  - The more negative, the less that we think the feature is associated with Spam email

# Intuition of the SGD algorithm on the "Spam Filter" example

$$\nabla \ell(w, (x_i, y_i)) = \frac{\exp(-y_i \cdot x_i^T w)}{1 + \exp(-y_i \cdot x_i^T w)} (-y_i x_i)$$

Scalar > 0:
≈ 0 if the prediction is correct
≈ 1 otherwise

Vector of dimension d: provides the direction of the gradient

If we receive an example [1, 0 , 0.0375, 80] like the one before.
And a label y = 1 saying that this is a spam.

How does the SGD update change the weight vector?

Then by moving w towards the negative gradient direction, we are changing the weight vector by increasing the weights. i.e., increasing the amount they contribute to the score function (if currently the classifier is making a mistake on this example)

# How to choose the step sizes / learning rates?

- In theory:
  - Gradient decent: 1/L where L is the **Gradient Lipschitz constant** of the function we minimize.
  - SGD: $\sum_t \eta_t = \infty, \sum_t \eta_t^2 < \infty$
    - e.g. $\eta_t \in [1/t, 1/\sqrt{t})$

$\sum_{t=1}^{n} \frac{1}{t} = 1 + \frac{1}{2} + \frac{1}{3} + \dots = O(\log n)$

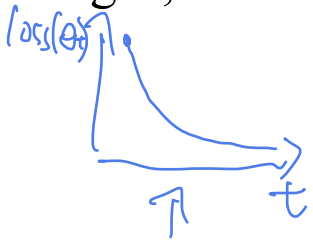
$\sum_{t=1}^{\infty} \frac{1}{t^2} = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \dots + \frac{1}{n^2} + \dots$

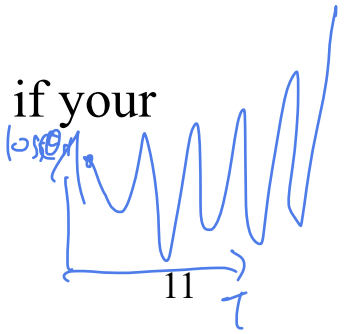
$= \frac{\pi^2}{6} \approx 1.65\dots < 2$

- In practice:
  - Use cross-validation on a subsample of the data.
  - Fixed learning rate for SGD is usually fine.
  - If it diverges, decrease the learning rate.
  - If for extremely small learning rate, it still diverges, check if your gradient implementation is correct.

validation

"learning curve"



# The power of SGD

- Extremely general:
  - Specify an end-to-end differentiable score function, e.g., a complex neural network.
  - Beyond the context of machine learning
- Extremely simple: a few lines of code.
- Extremely scalable
  - Just a few pass of the data, no need to store the data
- People are continuing to discover that many methods are special cases of SGD.

# Multiclass classification problem

- You will be implementing a multi-class classification algorithm using linear logistic regression.

$$\mathcal{Y} = \{1, 2, ..., k\}$$

- Recall that in binary classification you have one score function, the multi-class has one score function for each class.

$$h(x) = \underset{j \in \{1,...,k\}}{\text{argmax}} \, \text{Score}_{w_j}(x) = \underset{j \in \{1,...,k\}}{\text{argmax}} \, (x^T w_j)$$

- Now you have k linear score functions. Represent them as a matrix of weights W (what's the dimension? )
- You use the softmax-cross-entropy loss instead of the logistic loss.

# Softmax Cross-Entropy Loss in the binary classification case is equivalent to the Logistic Loss we learned

- Softmax cross entropy loss

$$\text{loss}(W)(x,y) = -\sum_{i=1}^{K} \mathbb{1}(y=i)\log(\hat{p}_i(x))$$

$$\hat{p}_i(x) = \text{softargmax}(\text{score}(W_i, x)) = \frac{\exp(W_i^T x)}{\sum_{j=1}^{K}\exp(W_j^T x)}$$

binary case: $-(y\cdot\log\hat{p} + (1-y)\cdot\log(1-\hat{p}))$

- In the binary case:

$$\ell(w) \quad \log(1+\exp(-y\cdot x^T w)) \qquad y\in\{-1,1\}$$

when $y=1$, $\ell(w) = -\log(1) + \log(1+\exp(-x^T w))$

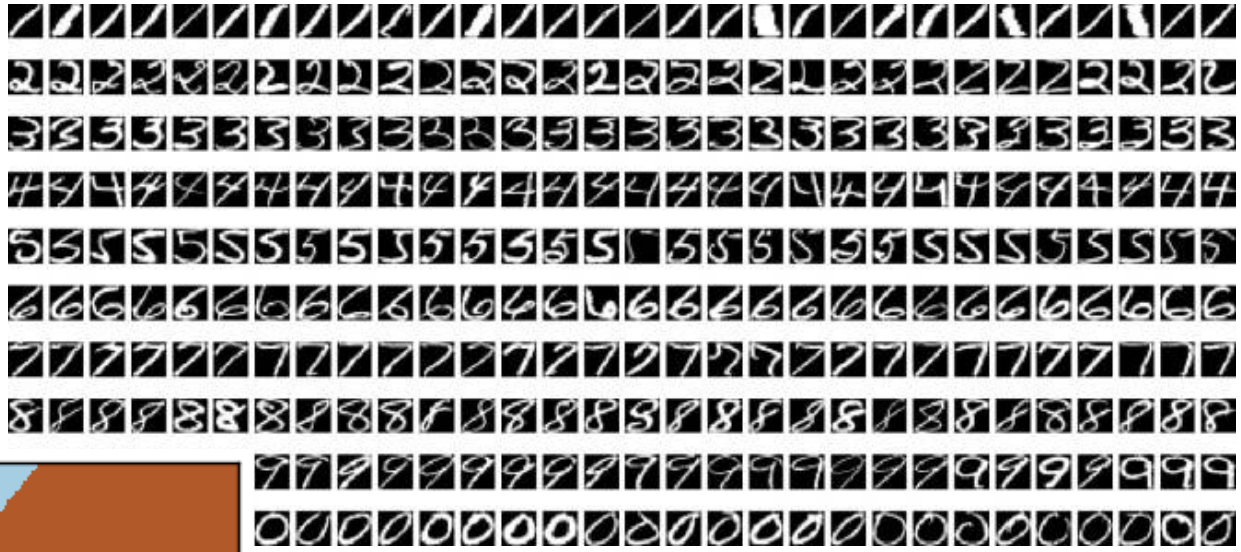
$$= -(\log(1) - \log(1+\exp(-x^T w)))$$

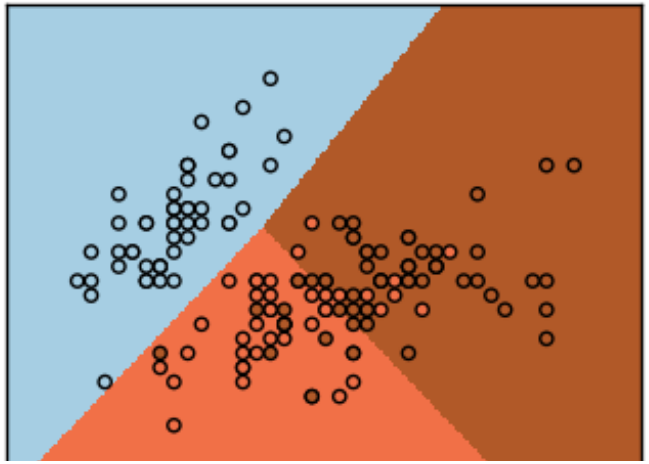
$$= -\log\left(\frac{1}{1+\exp(-x^T w)}\right) = -\log\left(\frac{\exp(x^T w)}{1+\exp(x^T w)}\right)$$

$\log x - \log y = \log\frac{x}{y}$ ?

when $y=-1$. (exercise)

# Decision boundaries of multi-class linear logistic regression





map image x to digit y

# From linear logistic regression to neural networks

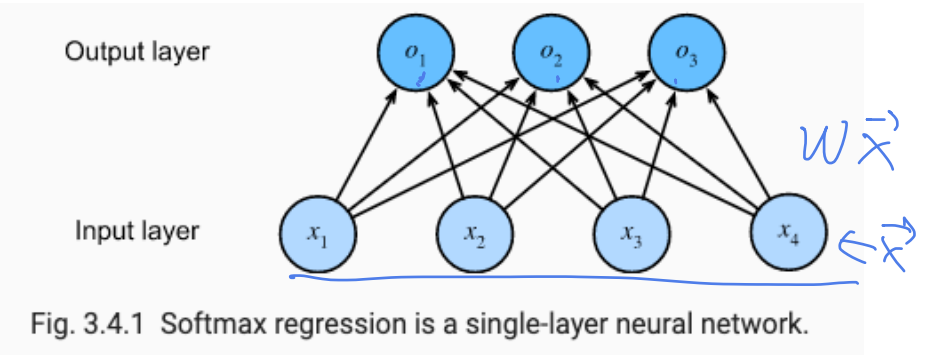



Fig. 3.4.1 Softmax regression is a single-layer neural network.

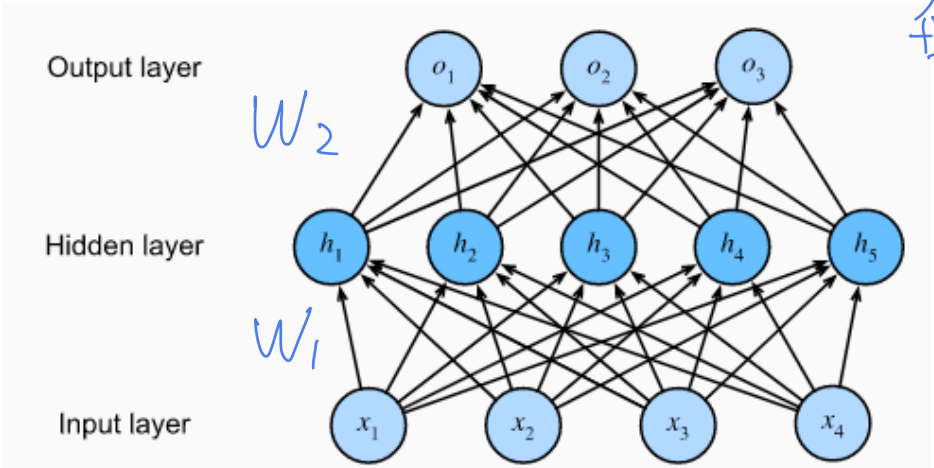



Fig. 4.1.1 An MLP with a hidden layer of 5 hidden units.

- Same input, same loss function
- The only difference is in how the score function is computed

# From linear logistic regression to neural networks (side by side comparison)

# Summary: steps to build a classifier agent

- 0. Collect a large labeled dataset.
- 1. Data splitting into training-validation-testing
- 2. Feature extraction
- 3. Specifying a hypothesis class
- 4. Learning with SGD using logistic loss
- 5. Validating on the validation set.
- 6. Testing on the test set.
- 7. Deploy the classifier agent.

Return to Step 2 or 3
If not happy with the validation error.
Do model selection (e.g. hyperparameter tuning)

# So far, we've learned everything about a classifier agent

- **Modeling**
  - Design meaningful feature extractors
  - Specify a family of classifiers on how the agent is going to classify examples using the features (indexed by free parameters)
- **Inference**: Trivially follows from the specification
- **Learning**: minimize errors (surrogate losses) over "free parameters"

This is called "discriminative" approach in modelling.

# "Generative" modeling: Modeling the world with a (joint) probability distribution

- The "Discriminative approach" doesn't care about the underlying processes that give rise to the observed data.

- **Modeling:** "Generative" modelling:
  - Label is a random variable Y, features are a vector random variables $X = [X_1, \ldots, X_d]^T$
  - Model the world with a (family of) joint probability distributions.

- **Inference**: Then we can make principled inference by calculating P(Y|X)
  - You don't just get a prediction, but also confidence.

- **Learning:** Find the distribution that fits the data well.

# Side-by-side comparisons of two modelling principles: "Discriminative" vs "Generative"

| | "Discriminative" | "Generative" |
|---|---|---|
| Modelling | • Extract features<br>• Hypothesis class<br>• $h_\theta: X \rightarrow Y$ | • How data are generated<br>• Family of probability distributions $P_\theta(X, Y)$ |
| Inference | Just apply the classifier: h(X) | Calculate P(Y\|X)<br><br>Predict with: $\max_y$P(Y=y \| X) |
| Learning | Minimize error over "free parameters": $\theta$ | Maximize likelihood over "free parameters": $\theta$ |

See Ng and Jordan (NIPS-2001):
https://ai.stanford.edu/~ang/papers/nips01-discriminativegenerative.pdf

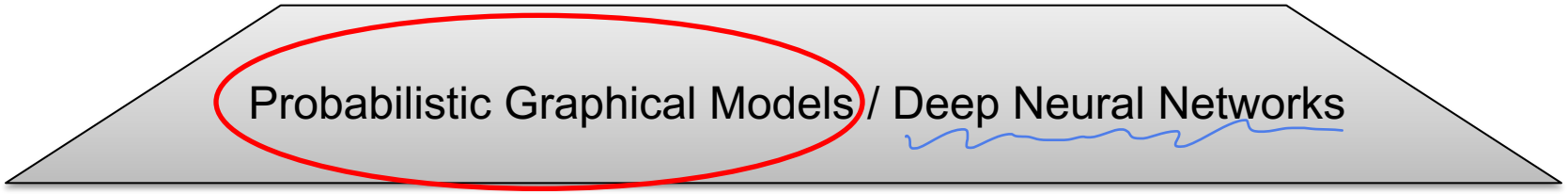
# Example: Spam filter

- "Discriminative modeling":
  - Think about features that has discriminative power
  - Think about hypothesis class (e.g., shape of the decision boundaries, functional form of the score function)

- "Generative modelling":
  - Think about how "spammers" write (generate) their emails.
  - And also how "non-spammers" write their emails.
  - Model the probability of certain words appear / the probability of certain word combinations appear.

Once we learned probabilistic graphical model, we will be able to do this effectively.

# Structure of the course




Classification / Regression
Bandits

Search
game playing

Markov Decision Processes
Reinforcement Learning

Logic, knowledge base
Probabilistic inference

**Reflex Agents**

**Planning Agents**

**Reasoning agents**

Low-level intelligence

High-level intelligence


(Again this idea is adapted from Percy Liang's teachings)

# Remaining time today and the next two lectures

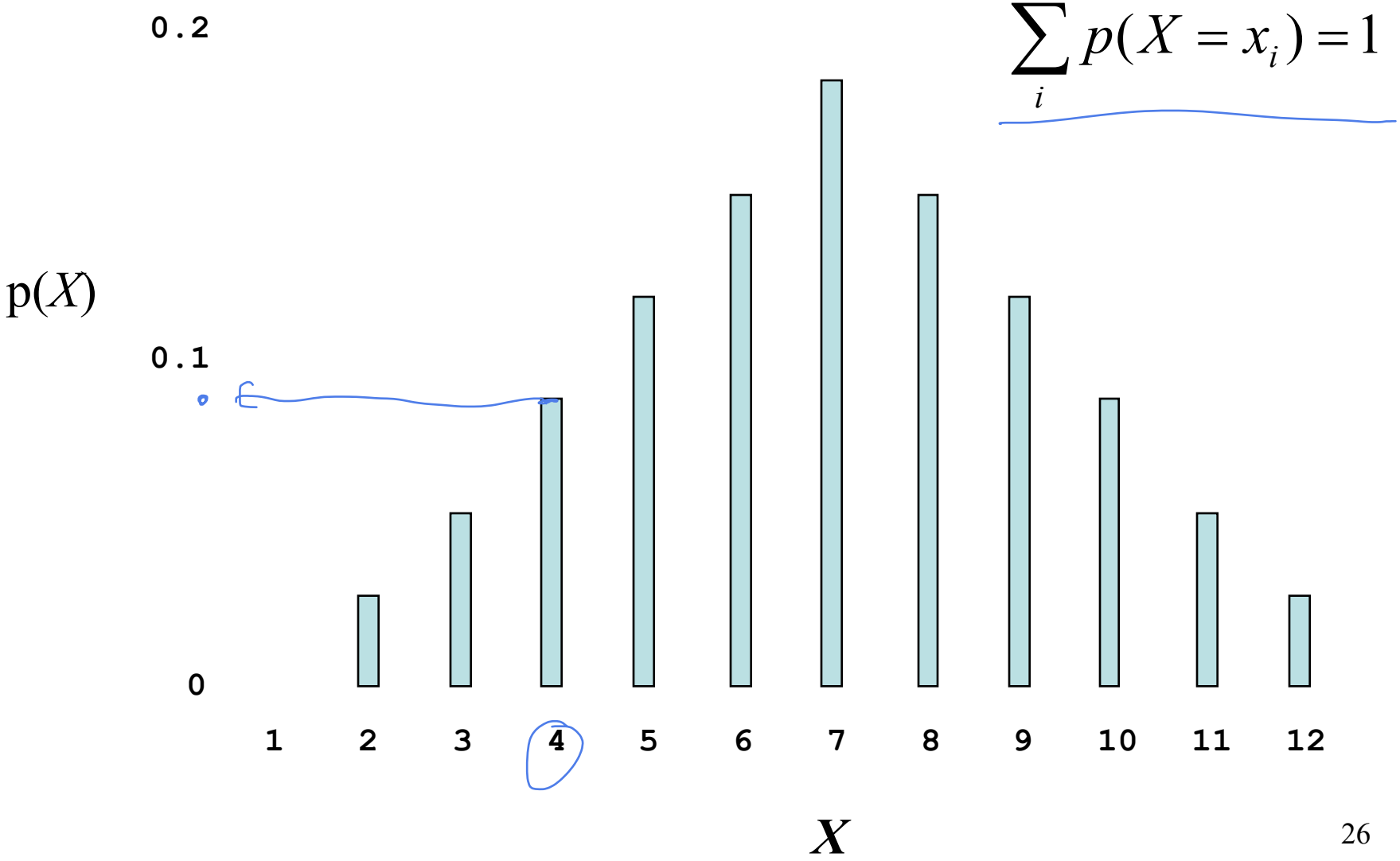
- Probability notations
- Joint distributions, marginal, conditional
- Representing these quantities as arrays / matrices
- Conditional independences
- BayesNet, and examples
- d-separation, reasoning and inference, probabilistic modelling

# Probability notation and notes

- Probabilities of *propositions*
  - P(A), P(the sun is shining)
- Probabilities of *random variables (r.v.)*
  - $P(X = x_1)$, $P(Y = y_1)$, $P(x_1 < X < x_2)$
- P(A) usually means P(A = True) (A is a proposition, not a variable)
  - This is a probability **value**
  - Technically, P(A) is a probability *function*
- $P(X = x_1)$
  - This is a probability **value** (P(X) is a probability *function*)
- P(X)
  - This is a **probability distribution** function, a.k.a probability mass function (**p.m.f.**) for discrete r.v. or a probability density function (**p.d.f.**) for continuous r.v.
- Technically, if X is an r.v., we should not write P(X) = 0.5
  - But rather $P(X = x_1) = 0.5$

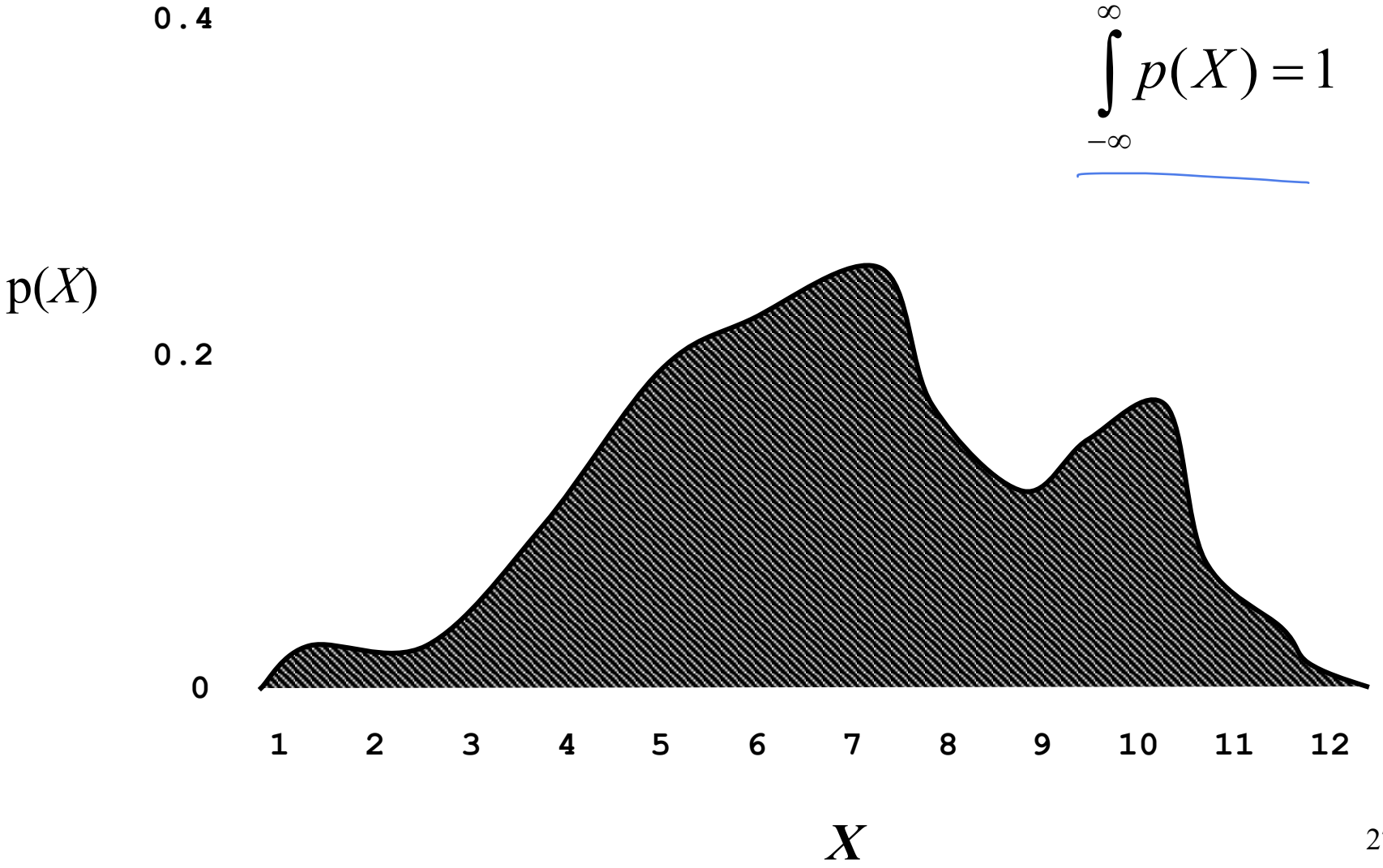
# Discrete probability distribution

$$\sum_i p(X = x_i) = 1$$



# Continuous probability distribution

$$\int_{-\infty}^{\infty} p(X) = 1$$



# Joint Probabilities

- A **complete probability model** is a single joint probability distribution over all propositions/variables in the domain
  - $P(X_1, X_2, \ldots, X_i, \ldots)$
- A particular instance of the world has the probability
  - $P(X_1=x_1 \wedge X_2=x_2 \wedge \ \ldots \wedge X_i=x_i \wedge \ \ldots) = p$
- Rather than stating knowledge as
  - Raining $\Rightarrow$ WetGrass
- We can state it as
  - P(Raining, WetGrass) = 0.15
  - P(Raining, ¬WetGrass) = 0.01
  - P(¬Raining, WetGrass) = 0.04
  - P(¬Raining, ¬WetGrass) = 0.8

| | ¬WetGrass | WetGrass |
|---|---|---|
| ¬Raining | 0.8 | 0.04 |
| Raining | 0.01 | 0.15 |

# Marginal and Conditional Probability

- Marginal Probability
  - Marginal probability (distribution) of X: $P(X) = \sum_Y P(X, Y)$
  - **Bayesian interpretation:** Probabilities associated with one proposition or variable, **prior** to any evidence
  - E.g., P(WetGrass), P(¬Raining)
- Conditional Probability
  - P(A | B) – "The probability of A given that we know B"
  - **Bayesian interpretation:** After (**posterior** to) procuring evidence
  - E.g., P(WetGrass | Raining)

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

Assumes P(Y) nonzero

or

$$P(X \mid Y)\, P(Y) = P(X, Y)$$

Conditional Marginal

# The chain rule: factorizing a joint distribution into marginal and conditionals

$$P(X,Y) = P(X \mid Y)\,P(Y)$$

By the Chain Rule

$$P(X,Y,Z) = P(X \mid Y,Z)P(Y,Z)$$
$$= P(X \mid Y,Z)\,P(Y \mid Z)\,P(Z)$$

*or, equivalently*

$$= P(X)P(Y \mid X)P(Z \mid X,Y)$$

Notes:

- Precedence: '|' is lowest
- E.g., P(X | Y, Z) means which?
  P( (X | Y), Z )
  P(X | (Y, Z) ) ⇐

# Chain Rule implies Bayes' Rule

**Thomas Bayes:** 1701 - 1761

- Since $P(X,Y) = P(X \mid Y)\,P(Y)$

  and $P(X,Y) = P(Y \mid X)\,P(X)$

- Then $P(X \mid Y)\,P(Y) = P(Y \mid X)\,P(X)$

conditional/likelihood, prior

$$P(X \mid Y) = \frac{P(Y \mid X)\,P(X)}{P(Y)} = \frac{P(Y \mid X)\,P(X)}{\sum_X P(Y \mid X)\,P(X)}$$

Bayes' Rule

normalization constant

**Funny fact**: Thomas Bayes is arguably a frequentist.

Stephen Fienberg. "When did Bayesian inference become 'Bayesian'?." *Bayesian analysis* 1.1 (2006): 1-40.
https://projecteuclid.org/euclid.ba/1340371071

# Representing Probability Distributions using linear algebraic data structures (in python)

| | Continuous vars | Discrete vars |
|---|---|---|
| P(X) | Function (of one variable) | m vector |
| P(X=x) | Scalar* | Scalar |
| P(X,Y) | Function of two variables | m×n matrix |
| P(X\|Y) | Function of two variables | m×n matrix |
| P(X\|Y=y) | Function of one variable | m vector |
| P(X=x\|Y) | Function of one variable | n vector |
| P(X=x\|Y=y) | Scalar* | Scalar |

* - actually zero. Should be $P(x_1 < X < x_2)$

# Example: Joint probability distribution

From P(X,Y), we can always calculate:

| | |
|---|---|
| P(X) | $P(X=x_1)$ |
| P(Y) | $P(Y=y_2)$ |
| P(X\|Y) | $P(X\|Y=y_1)$ |
| P(Y\|X) | $P(Y\|X=x_1)$ |
| | $P(X=x_1\|Y)$ |
| | etc. |

| **Y** \ **X** | $x_1$ | $x_2$ | $x_3$ |
|---|---|---|---|
| $y_1$ | 0.2 | 0.1 | 0.1 |
| $y_2$ | 0.1 | 0.2 | ? |

**P(X,Y)**

| | $x_1$ | $x_2$ | $x_3$ |
|---|---|---|---|
| $y_1$ | 0.2 | 0.1 | 0.1 |
| $y_2$ | 0.1 | 0.2 | 0.3 |

**P(X)**

| $x_1$ | $x_2$ | $x_3$ |
|---|---|---|
| 0.3 | 0.3 | 0.4 |

**P(Y)**

| | |
|---|---|
| $y_1$ | 0.4 |
| $y_2$ | 0.6 |

**P(X|Y)**

| | $x_1$ | $x_2$ | $x_3$ |
|---|---|---|---|
| $y_1$ | 0.5 | 0.25 | 0.25 |
| $y_2$ | 0.167 | 0.333 | 0.5 |

$P(X=x_1, Y=y_2) = ?$

$P(X=x_1) = ?$

$P(Y=y_2) = ?$

$P(X|Y=y_1) = ?$

$P(X=x_1|Y) = ?$

**P(Y|X)**

| | $x_1$ | $x_2$ | $x_3$ |
|---|---|---|---|
| $y_1$ | 0.667 | 0.333 | 0.25 |
| $y_2$ | 0.333 | 0.667 | 0.75 |

# Quick checkpoint

- Probability notations
  - P(A) is a number when A is an event / predicate.
  - P(X) is a vector/function when X is a random variable.

- Joint probability distribution
  - Enumerating all combinations of events.
  - All values the random variables can take.
  - Assign a non-negative value to each.

- Marginals, conditionals
  - How they are related: Chain rule, Bayes rule