Lecture 17

Amit Kumar Das

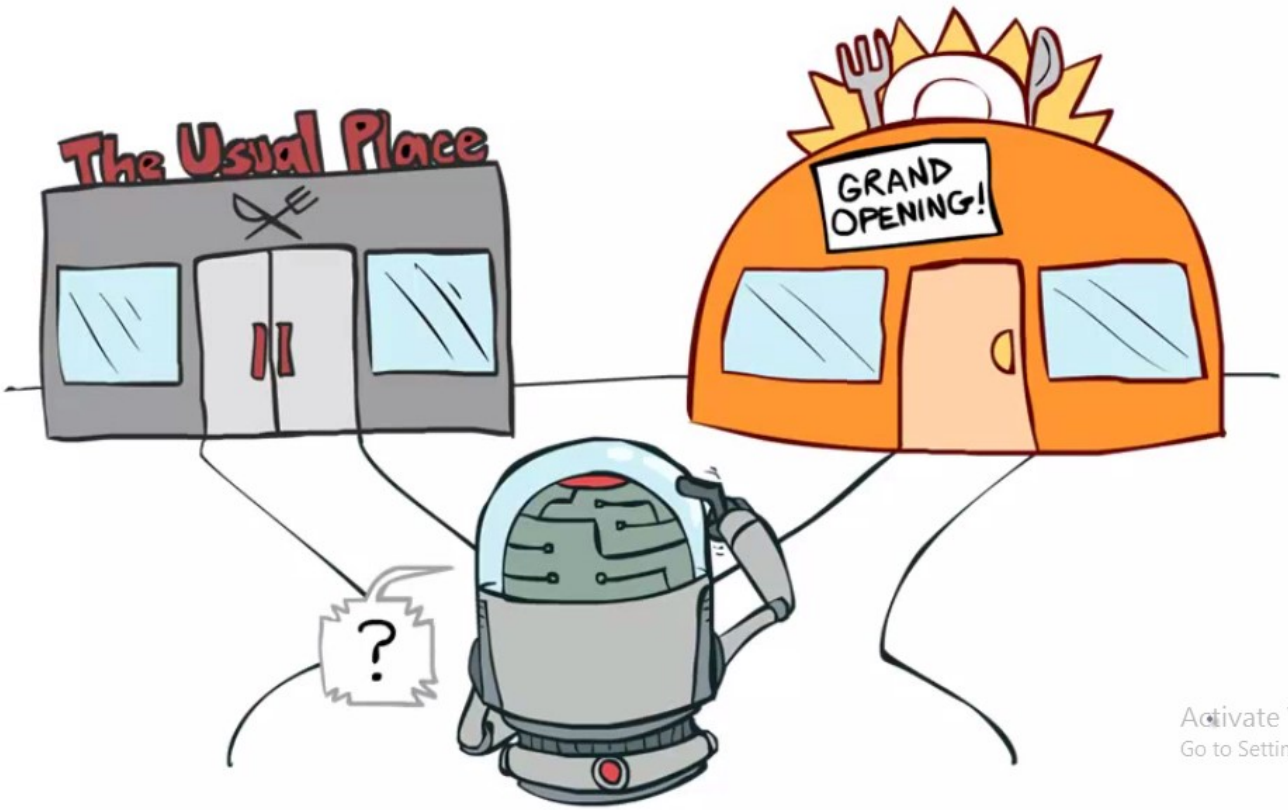
Senior Lecturer,

Department of Computer Science & Engineering,

East West University

Dhaka, Bangladesh.

Exploration vs. Exploitation



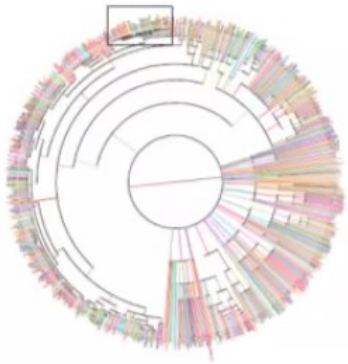
Exploration vs. exploitation

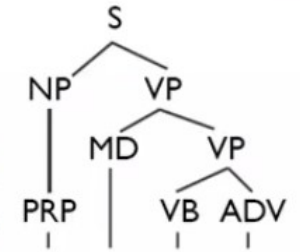
- **Exploration:** take a new action with unknown consequences
  - Pros:
    - Get a more accurate model of the environment
    - Discover higher-reward states than the ones found so far
  - Cons:
    - When you're exploring, you're not maximizing your utility
    - Something bad might happen
- **Exploitation:** go with the best strategy found so far
  - Pros:
    - Maximize reward as reflected in the current utility estimates
    - Avoid bad stuff
  - Cons:
    - Might also prevent you from discovering the true optimal strategy

How to Explore?

- Several schemes for forcing exploration
  - Simplest: random actions (ε-greedy)
    - Every time step, flip a coin
    - With (small) probability ε, act randomly
    - With (large) probability 1-ε, act on current policy
  - Problems with random actions?
    - You do eventually explore the space, but keep thrashing around once learning is done
    - One solution: lower ε over time
    - Another solution: exploration functions

Now: Advanced Applications





You will see later

Después lo veras

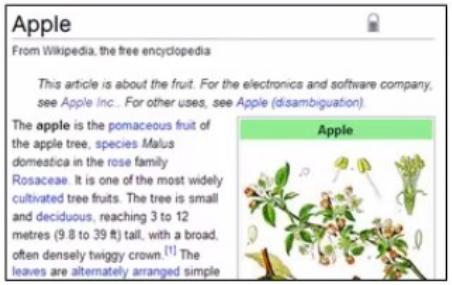
Web Search

Extension: Web Search

- Information retrieval:
  - Given information needs, produce information
  - Includes, e.g. web search, question answering, and classic IR

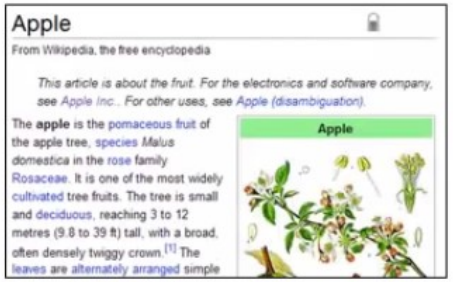
- Web search: not exactly classification, but rather ranking

x = “Apple Computers”



Feature-Based Ranking

x = "Apple Computer"

$f($ $x,$ 

$) = [0.3\ 5\ 0\ 0\ \ldots]$

$f($ $x,$ 

$) = [0.8\ 4\ 2\ 1\ \ldots]$

What is NLP?


What we say to dogs
Okay, Ginger! I've had it! You stay out of the garbage! Understand, Ginger? Stay out of the garbage, or else!

What is NLP?


What we say to dogs
Okay, Ginger! I've had it! You stay out of the garbage! Understand, Ginger? Stay out of the garbage, or else!


What they hear
blah blah GINGER blah blah blah blah blah blah blah GINGER blah blah blah blah blah...

What is NLP?

- Fundamental goal: analyze and process human language, broadly, robustly, accurately…
- End systems that we want to build:
  - Ambitious: speech recognition, machine translation, information extraction, dialog interfaces, question answering…
  - Modest: spelling correction, text categorization…

Problem: Ambiguities

- Headlines:
  - Enraged Cow Injures Farmer With Ax

Problem: Ambiguities

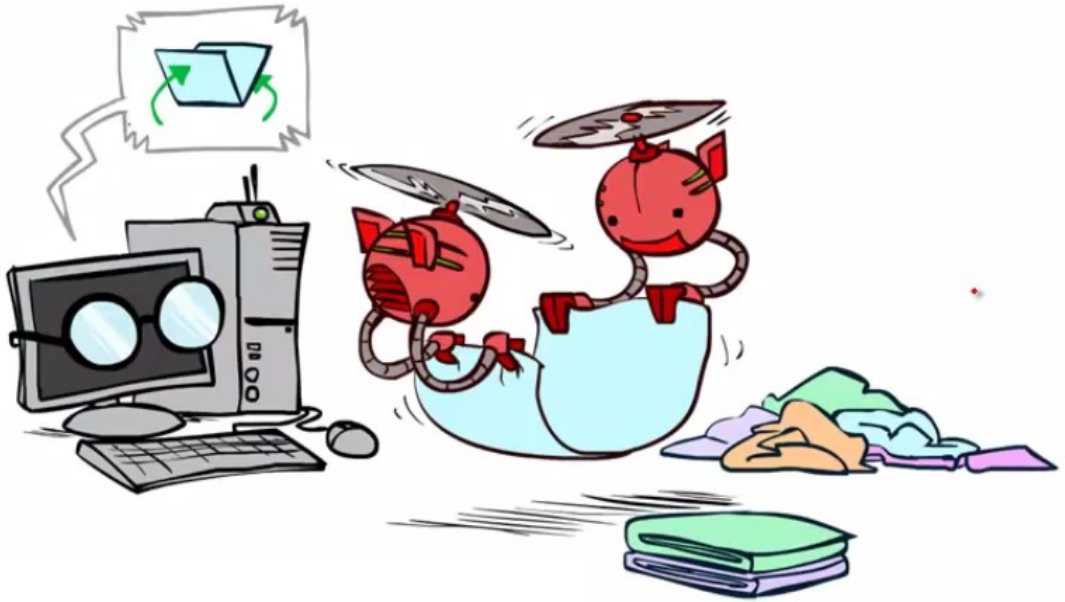
- Headlines:
  - Enraged Cow Injures Farmer With Ax

Machine Translation

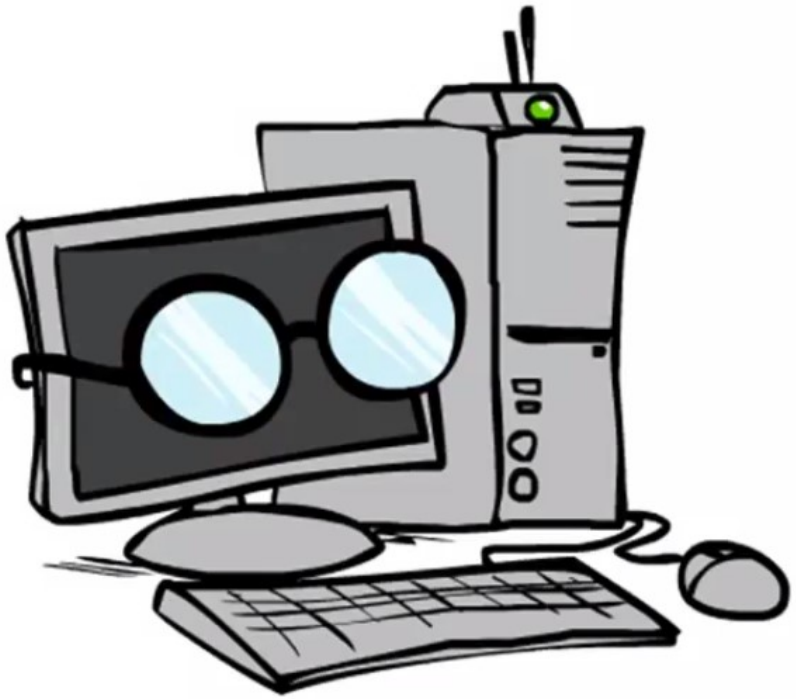
Machine Translation

- Translate text from one language to another
- Recombines fragments of example translations
- Challenges:
  - What fragments? [learning to translate]
  - How to make efficient? [fast translation search]

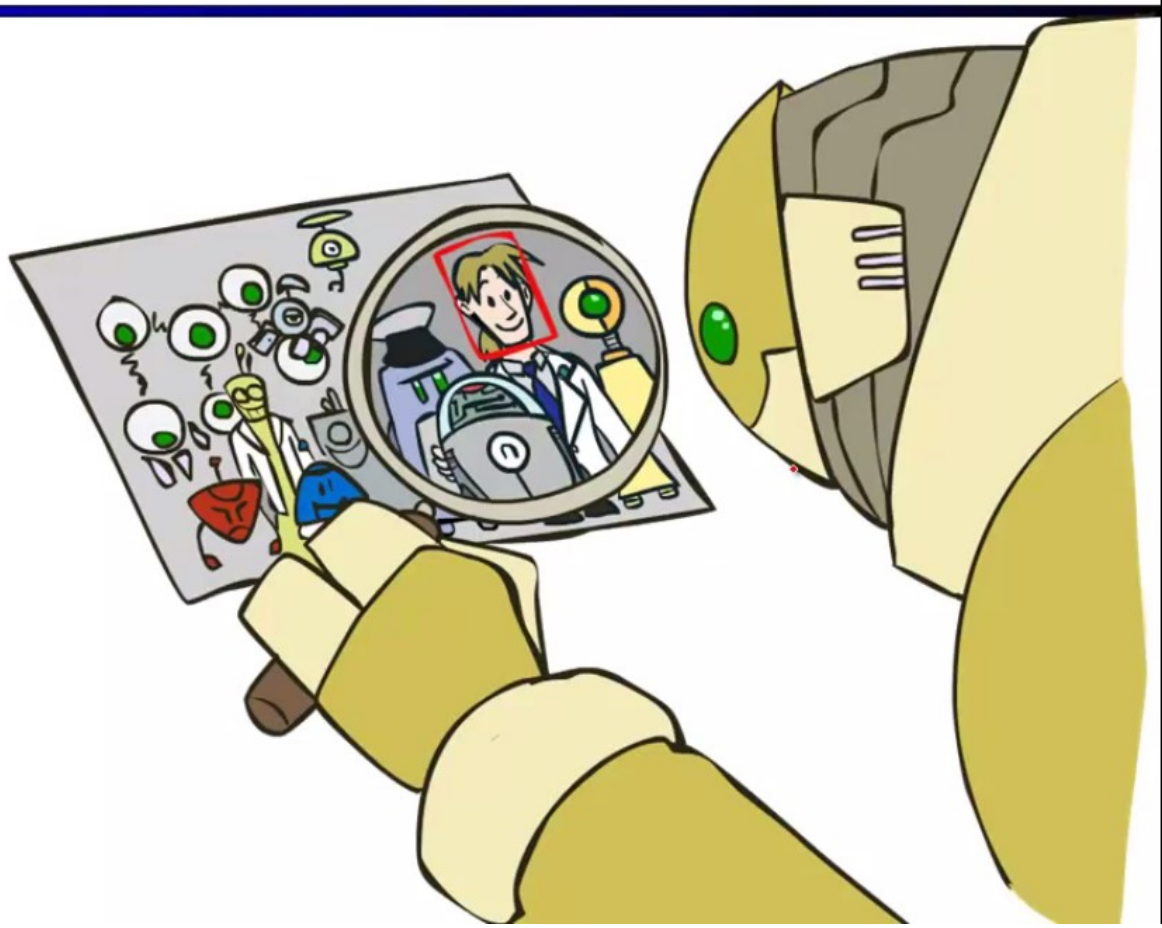
Advanced Applications: Computer Vision and Robotics *



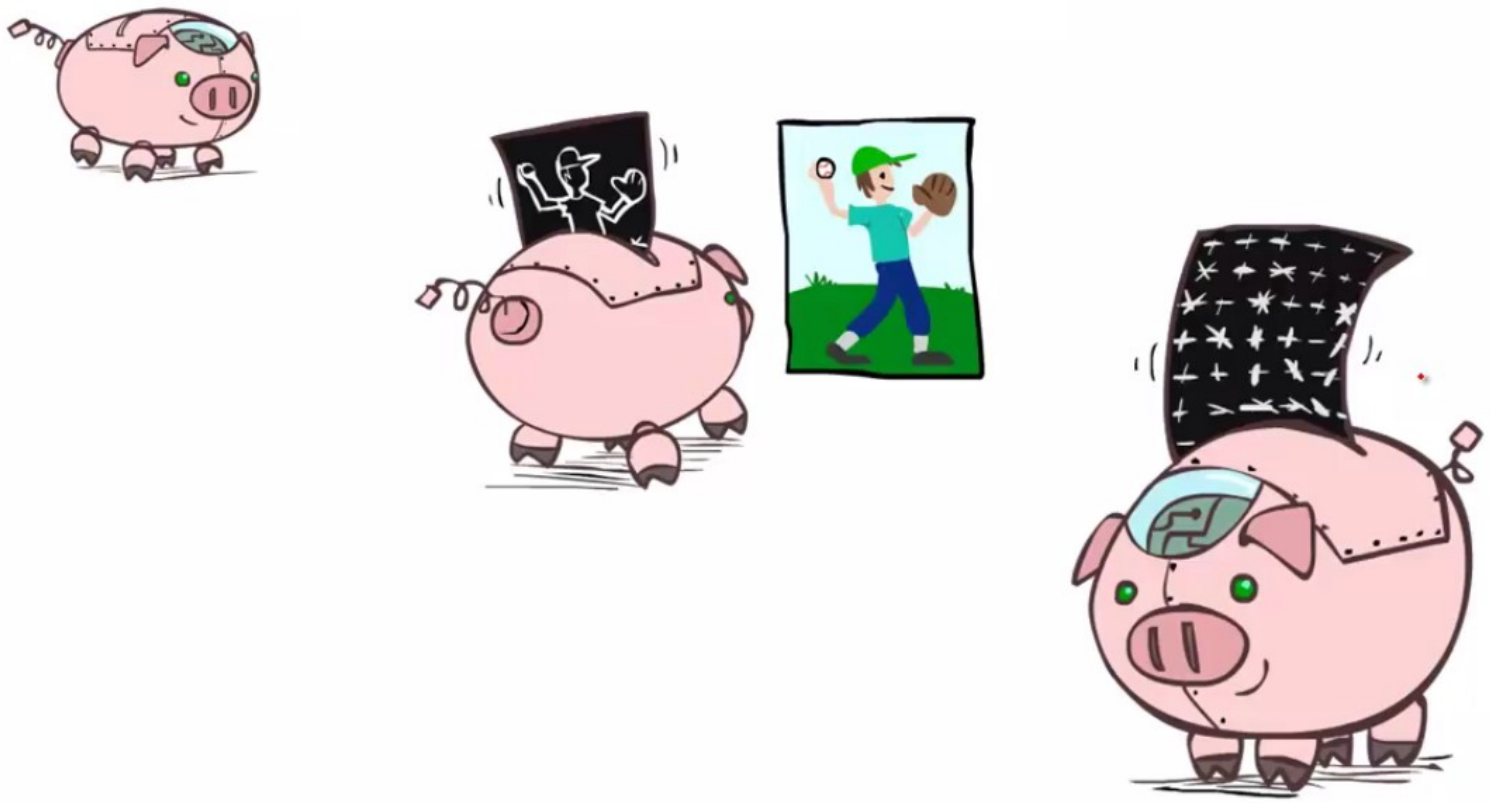
Computer Vision



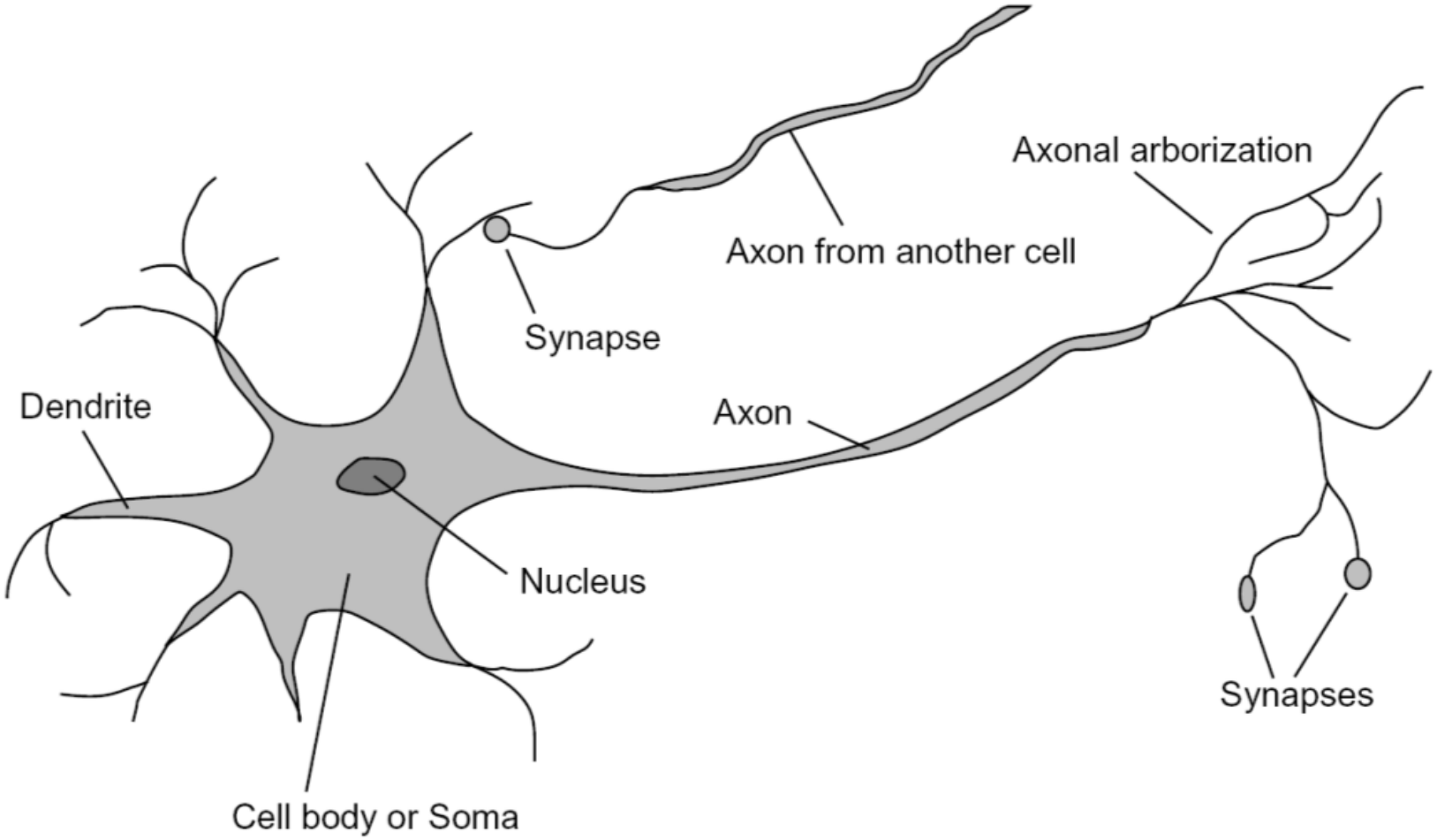
Object Detection



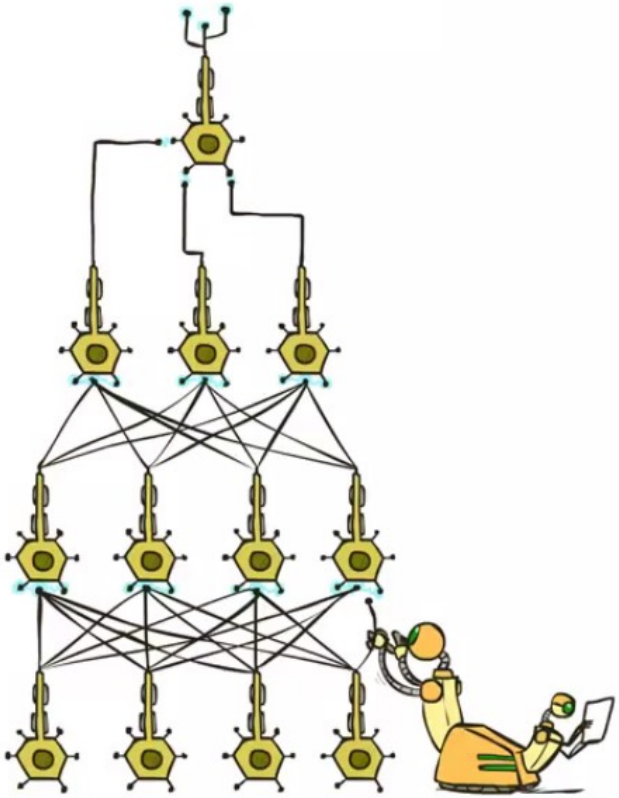
Object Detection Approach 1: HOG + SVM



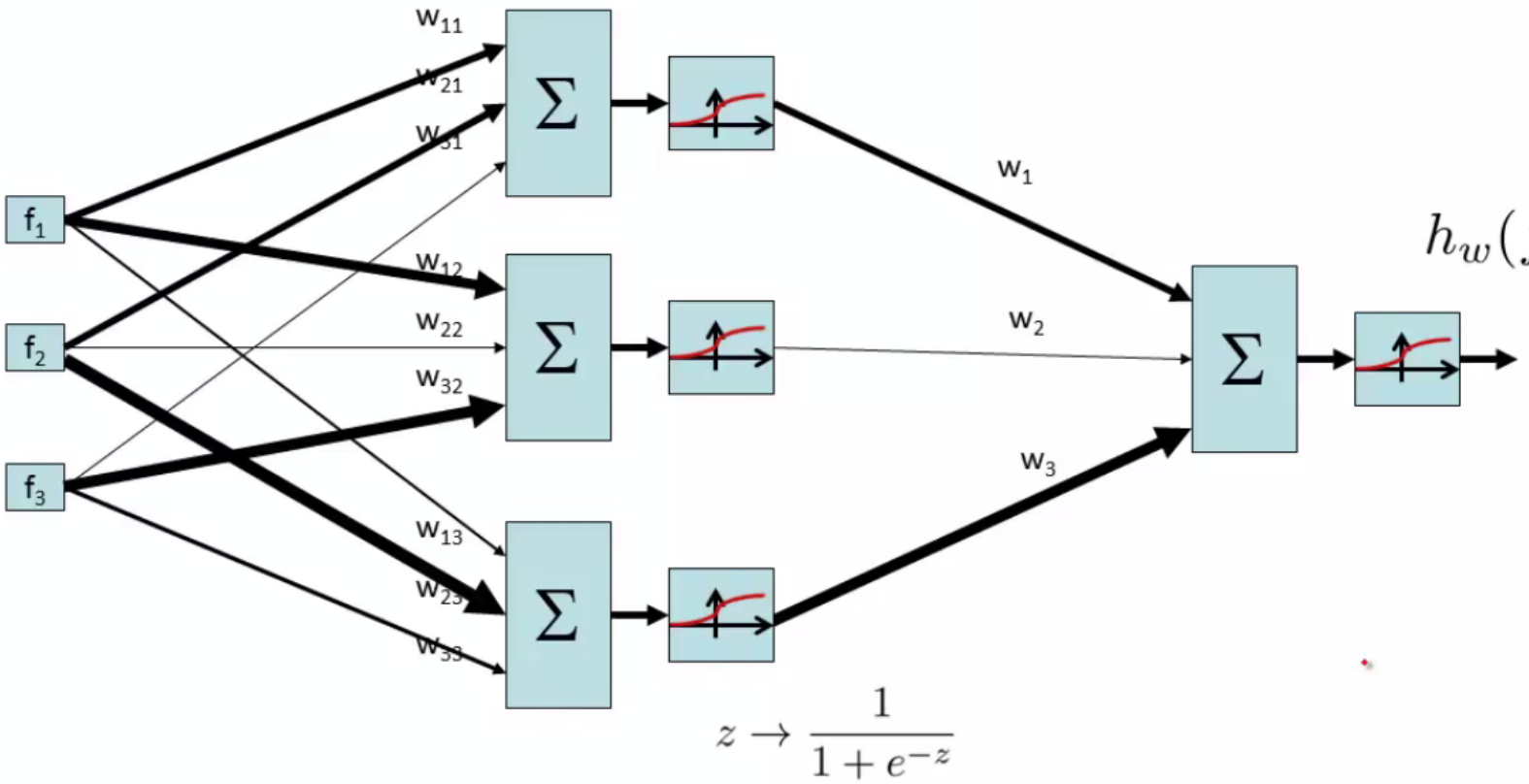
Loose inspiration: Human neurons



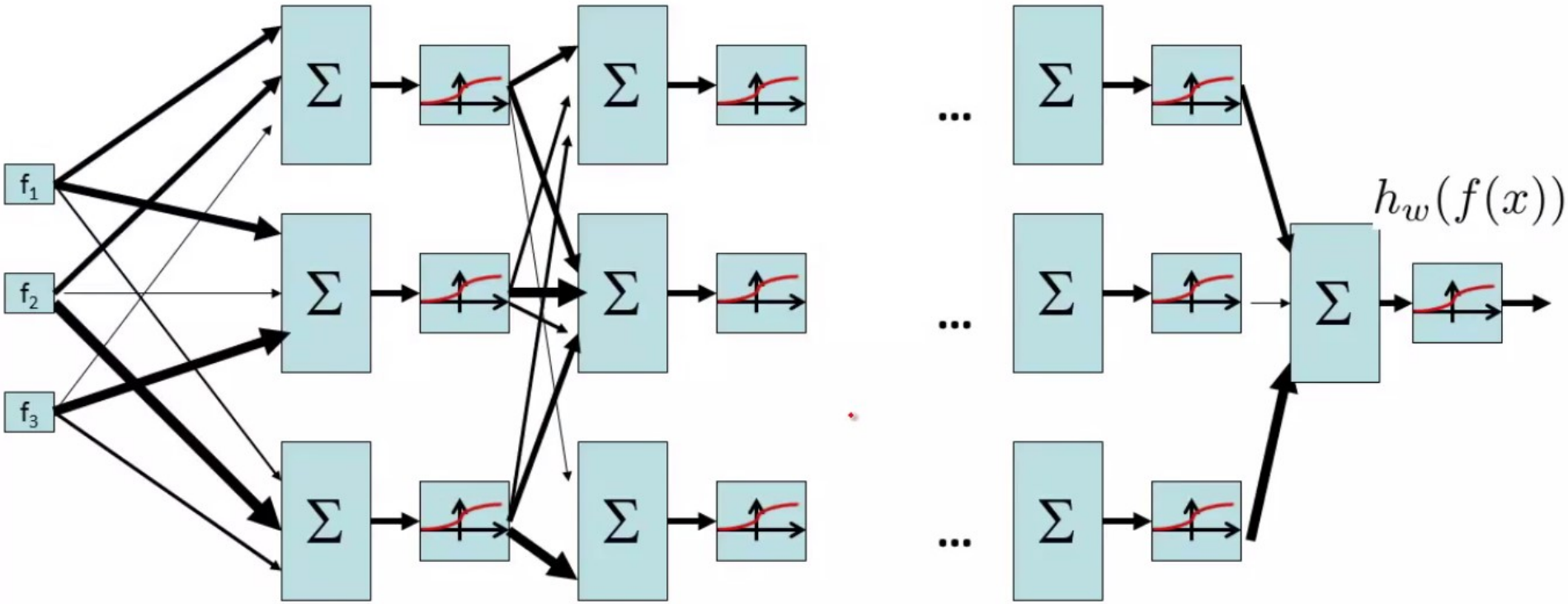
Object Detection Approach 2: Deep Learning



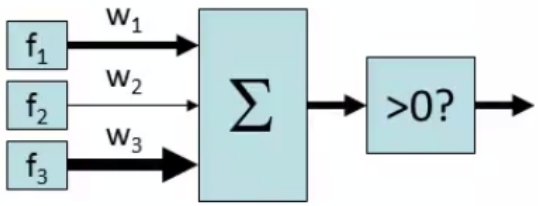
Two-Layer Neural Network



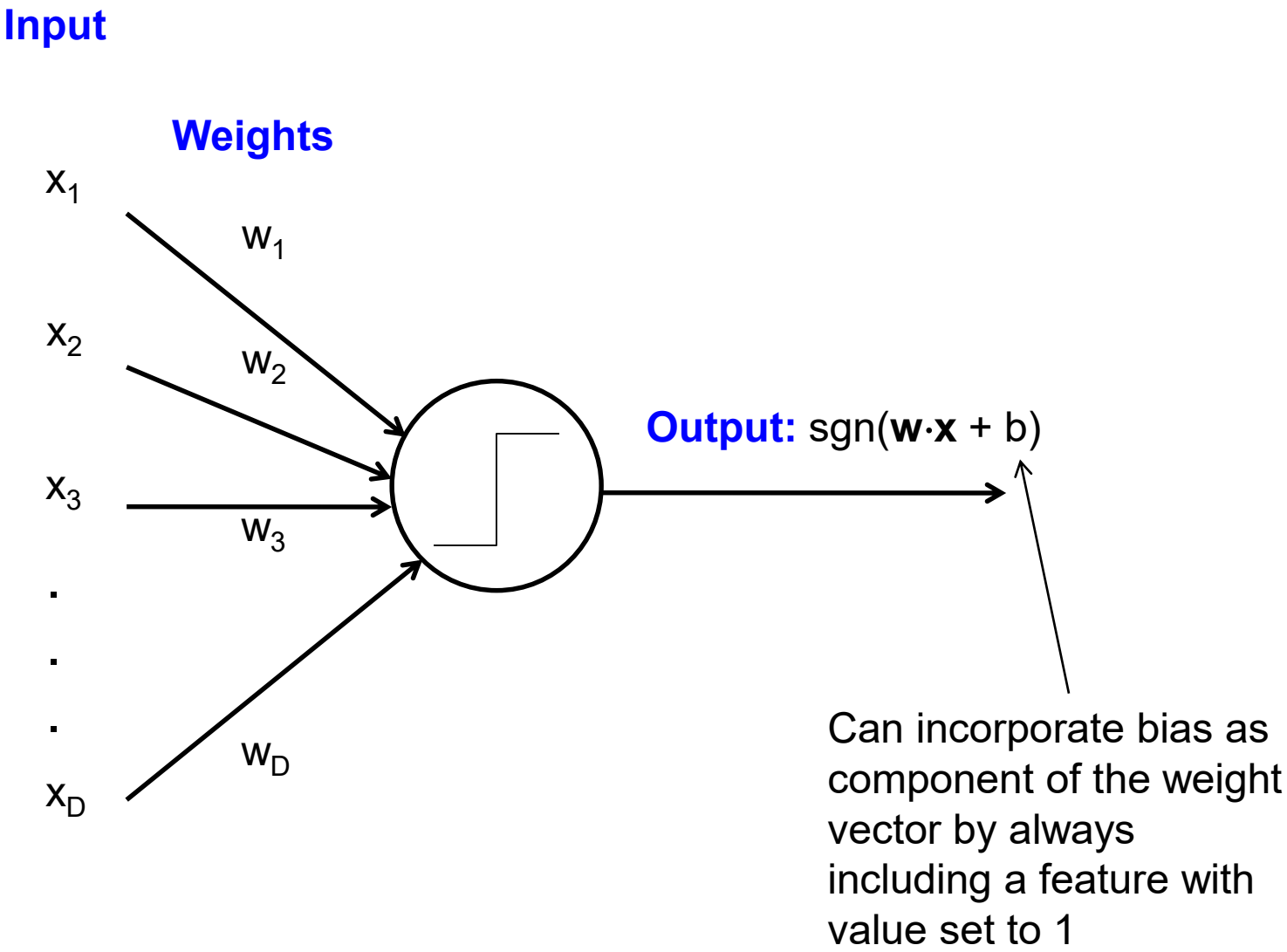
N-Layer Neural Network



Perceptron



Perceptron


Input
Weights
x_1
x_2
x_3
x_D
w_1
w_2
w_3
w_D
Output: sgn(**w**·**x** + b)
Can incorporate bias as component of the weight vector by always including a feature with value set to 1

Perceptron

- Current weight vector: w1=(1, 2, -2), w2 = (3, -2, -1), w3 = (-1, 2, 4)
- Next training data point: f(x) = (1, -0.5, 3), y = 2
- Perceptron update:

Perceptron

- Current weight vector: w1=(1, 2, -2), w2 = (3, -2, -1), w3 = (-1, 2, 4)
- Next training data point: f(x) = (1, -0.5, 3), y = 2
- Perceptron update:

$$w_1 \cdot f(x) = 1 \cdot 1 + 2 \cdot (-0.5) + (-2) \cdot 3 = -6$$

$$w_2 \cdot f(x) = 3 \cdot 1 - 2 \cdot (-0.5) - 1 \cdot 3 = 1$$

$$w_3 \cdot f(x) = -1 \cdot 1 + 2 \cdot (-0.5) + 4 \cdot 3 = 10$$

Perceptron

- Current weight vector: w1=(1, 2, -2), w2 = (3, -2, -1), w3 = (-1, 2, 4)
- Next training data point: f(x) = (1, -0.5, 3), y = 2
- Perceptron update:

$w_1 \cdot f(x) = 1 \cdot 1 + 2 \cdot (-0.5) + (-2) \cdot 3 = -6$

$w_2 \cdot f(x) = 3 \cdot 1 - 2 \cdot (-0.5) - 1 \cdot 3 = 1$

$w_3 \cdot f(x) = -1 \cdot 1 + 2 \cdot (-0.5) + 4 \cdot 3 = 10$

$\rightarrow$ prediction: $y = 3$

Perceptron

- Current weight vector: w1=(1, 2, -2), w2 = (3, -2, -1), w3 = (-1, 2, 4)
- Next training data point: f(x) = (1, -0.5, 3), y* = 2
- Perceptron update:

$w_1 \cdot f(x) = 1 \cdot 1 + 2 \cdot (-0.5) + (-2) \cdot 3 = -6$

$w_2 \cdot f(x) = 3 \cdot 1 - 2 \cdot (-0.5) - 1 \cdot 3 = 1$

$w_3 \cdot f(x) = -1 \cdot 1 + 2 \cdot (-0.5) + 4 \cdot 3 = 10$

→ prediction: $y = 3$

$w_2 \leftarrow w_2 + f(x)$

$(3, -2, -1) + (1, -0.5, 3) = (4, -2.5, 2)$

Perceptron training algorithm

- Initialize weights
- Cycle through training examples in multiple passes (*epochs*)
- For each training example:
  - If classified correctly, do nothing
  - If classified incorrectly, update weights

Perceptron update rule

- For each training instance **x** with label y:
  - Classify with current weights: $y' = \text{sgn}(\mathbf{w} \cdot \mathbf{x})$
  - Update weights: $\mathbf{w} \leftarrow \mathbf{w} + \alpha(y-y')\mathbf{x}$
  - α is a learning rate that should decay as a function of epoch t, e.g., 1000/(1000+t)
  - What happens if y' is correct?
  - Otherwise, consider what happens to individual weights $w_i \leftarrow w_i + \alpha(y-y')x_i$
    - If y = 1 and y' = -1, w_i will be increased if x_i is positive or decreased if x_i is negative → **w**·**x** will get bigger
    - If y = -1 and y' = 1, w_i will be decreased if x_i is positive or increased if x_i is negative → **w**·**x** will get smaller

Convergence of perceptron update rule

- **Linearly separable data:** converges to a perfect solution
- **Non-separable data:** converges to a minimum-error solution assuming learning rate decays as O(1/t) and examples are presented in random sequence

Implementation details

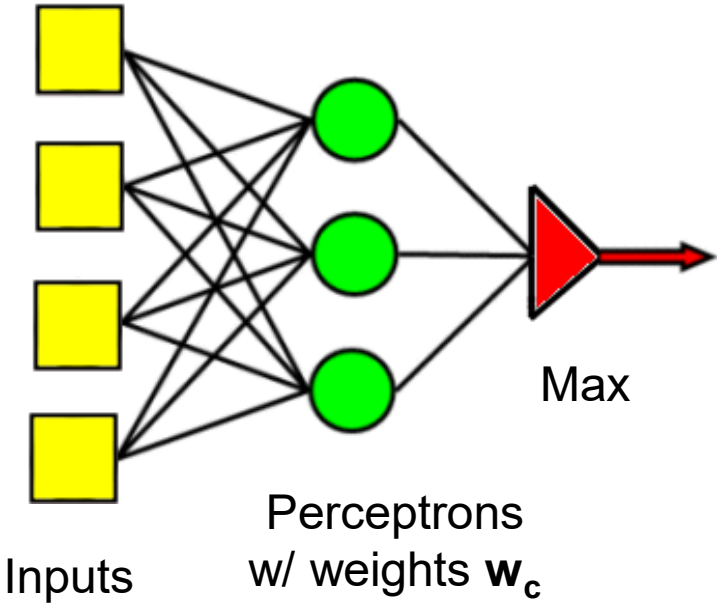
- Bias (add feature dimension with value fixed to 1) vs. no bias
- Initialization of weights: all zeros vs. random
- Learning rate decay function
- Number of epochs (passes through the training data)
- Order of cycling through training examples (random)

Multi-class perceptrons

- *One-vs-others* framework: Need to keep a weight vector $\mathbf{w}_c$ for each class c
- Decision rule: $c = \text{argmax}_c \, \mathbf{w}_c \cdot \mathbf{x}$
- Update rule: suppose example from class c gets misclassified as c'
  - Update for c: $\mathbf{w}_c \leftarrow \mathbf{w}_c + \alpha \mathbf{x}$
  - Update for c': $\mathbf{w}_{c'} \leftarrow \mathbf{w}_{c'} - \alpha \mathbf{x}$

Review: Multi-class perceptrons

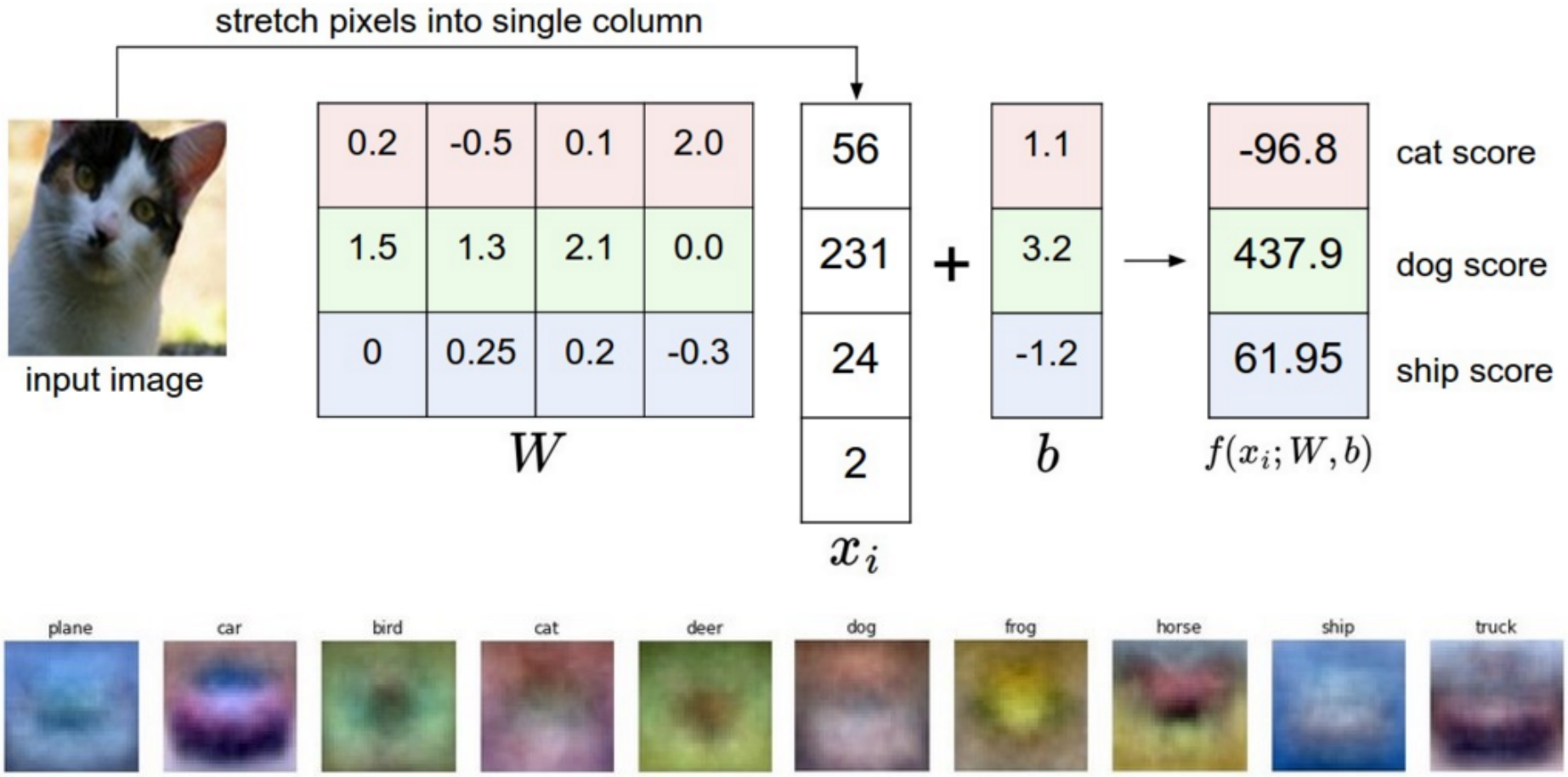
- *One-vs-others* framework: Need to keep a weight vector $\mathbf{w}_c$ for each class c
- Decision rule: $c = \text{argmax}_c \, \mathbf{w}_c \cdot \mathbf{x}$



Softmax:

$$P(c \mid \mathbf{x}) = \frac{\exp(\mathbf{w}_c \cdot \mathbf{x})}{\sum_{k=1}^{C} \exp(\mathbf{w}_k \cdot \mathbf{x})}$$

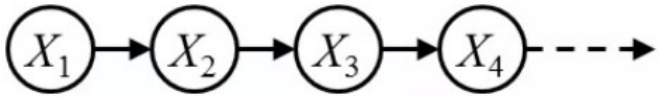
Visualizing linear classifiers



Source: Andrej Karpathy, http://cs231n.github.io/linear-classify/

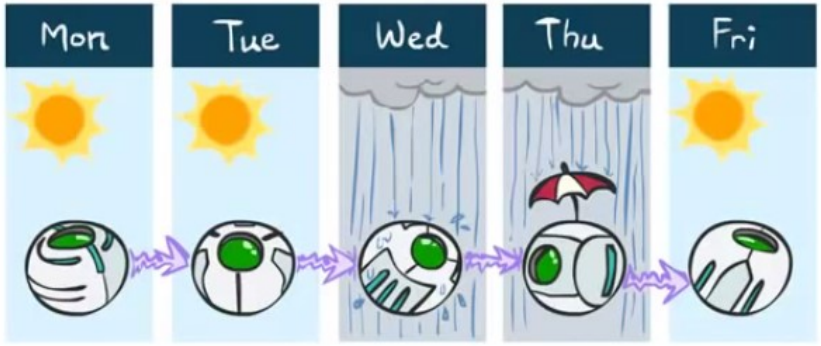
Markov Models

- Value of X at a given time is called the state



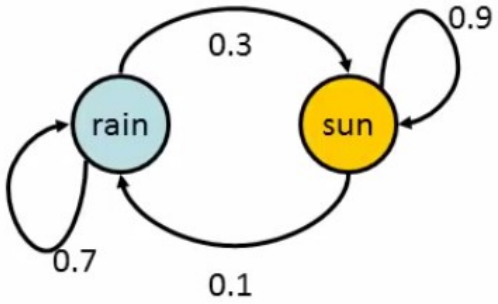
Example Markov Chain: Weather

- States: X = {rain, sun}
- Initial distribution: 1.0 sun
- CPT $P(X_t \mid X_{t-1})$:



X_{t-1}	X_t	$P(X_t \mid X_{t-1})$
sun	sun	0.9
sun	rain	0.1
rain	sun	0.3
rain	rain	0.7

Two new ways of representing the same CPT

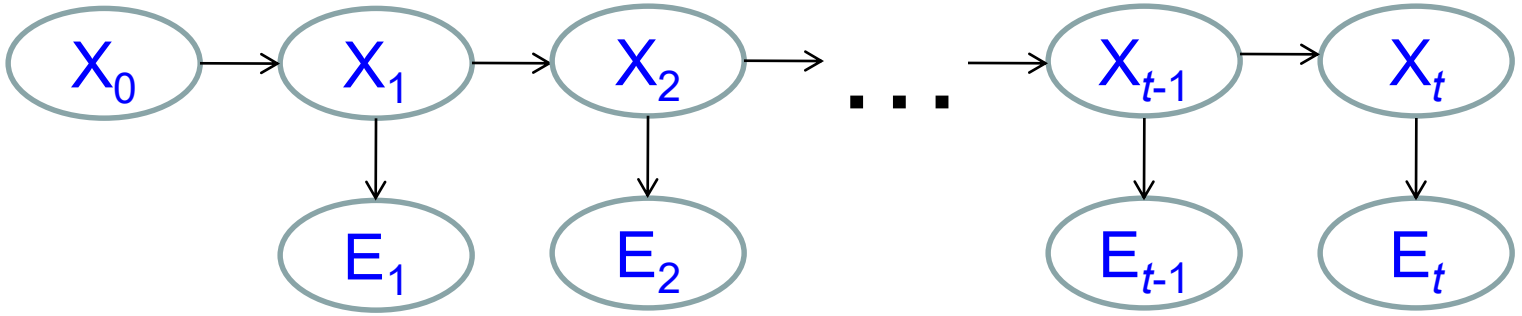


Probabilistic reasoning over time

- So far, we've mostly dealt with *episodic* environments
  - Exceptions: games with multiple moves, planning
- In particular, the Bayesian networks we've seen so far describe static situations
  - Each random variable gets a single fixed value in a single problem instance
- Now we consider the problem of describing probabilistic environments that evolve over time
  - Examples: robot localization, tracking, speech, …

Hidden Markov Models

- At each time slice t, the state of the world is described by an unobservable variable X_t and an observable *evidence* variable E_t
- **Transition model:** distribution over the current state given the whole past history: $P(X_t \mid X_0, \ldots, X_{t-1}) = P(X_t \mid \mathbf{X}_{0:t-1})$
- **Observation model:** $P(E_t \mid \mathbf{X}_{0:t}, \mathbf{E}_{1:t-1})$

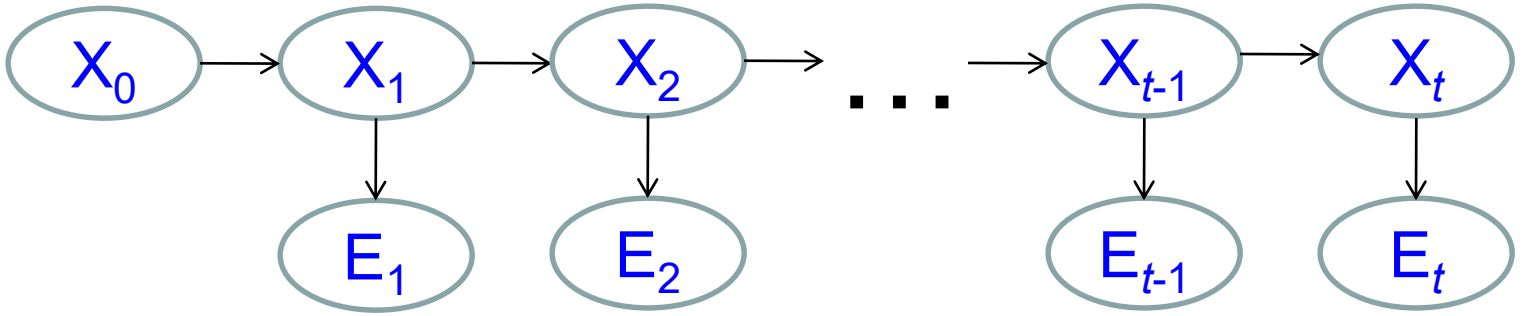


Hidden Markov Models

- **Markov assumption** (first order)
  - The current state is conditionally independent of all the other states given the state in the previous time step
  - What does $P(X_t \mid \mathbf{X}_{0:t-1})$ simplify to?

    $P(X_t \mid \mathbf{X}_{0:t-1}) = P(X_t \mid X_{t-1})$
- Markov assumption for observations
  - The evidence at time *t* depends only on the state at time *t*
  - What does $P(E_t \mid \mathbf{X}_{0:t}, \mathbf{E}_{1:t-1})$ simplify to?

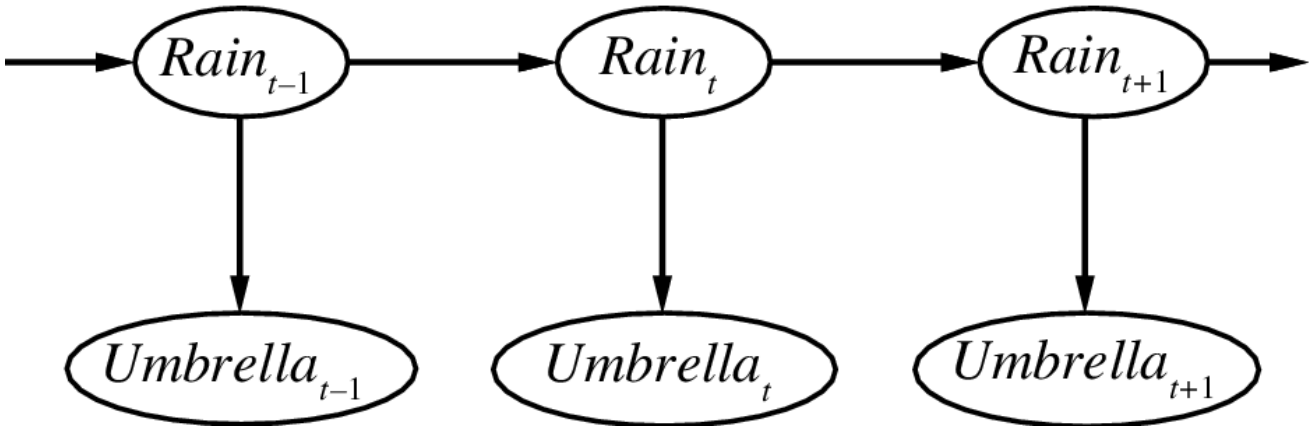
    $P(E_t \mid \mathbf{X}_{0:t}, \mathbf{E}_{1:t-1}) = P(E_t \mid X_t)$



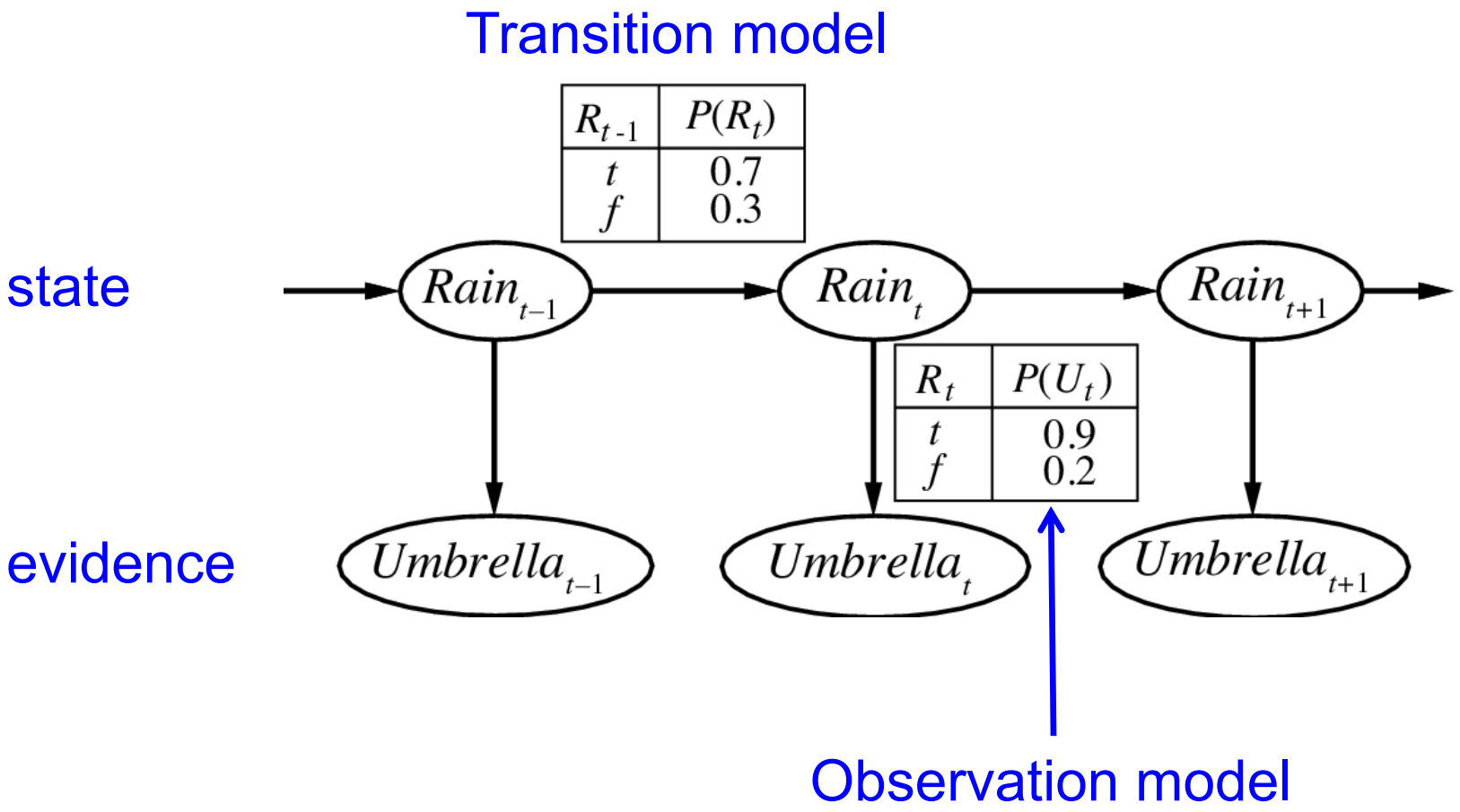
Example

state

evidence

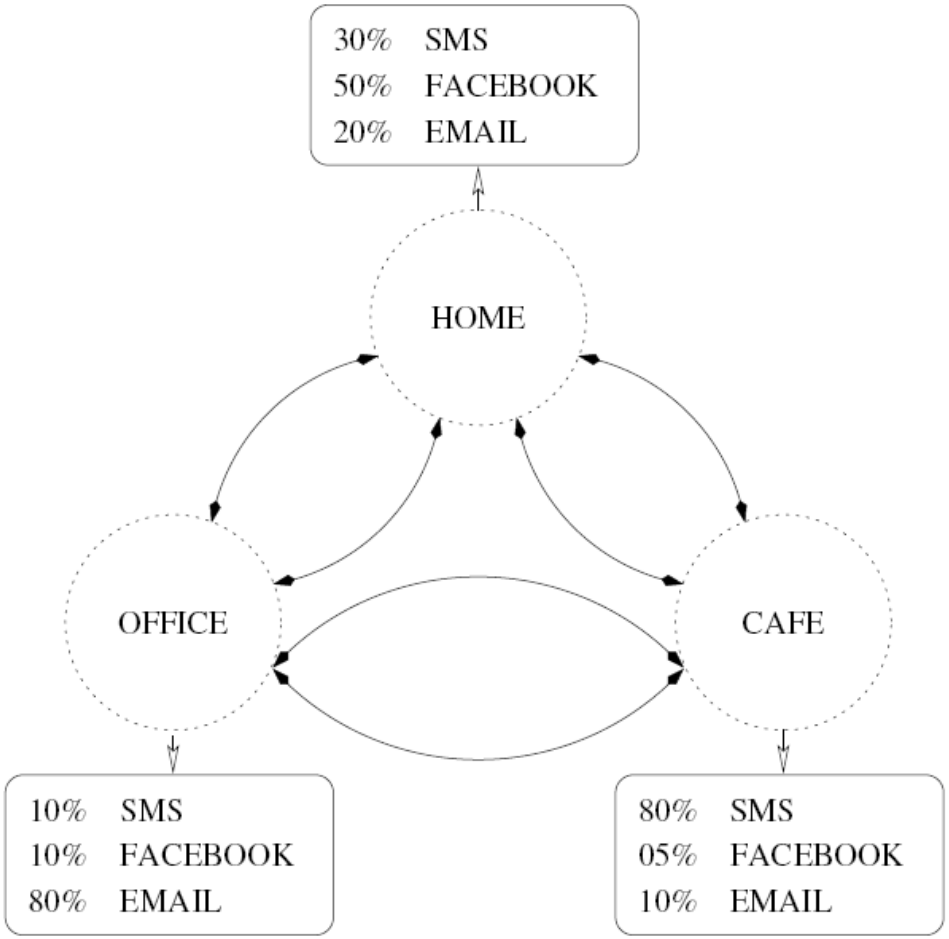


Example


Transition model
R_{t-1} | $P(R_t)$
t | 0.7
f | 0.3
state
$Rain_{t-1}$
$Rain_t$
$Rain_{t+1}$
R_t | $P(U_t)$
t | 0.9
f | 0.2
evidence
$Umbrella_{t-1}$
$Umbrella_t$
$Umbrella_{t+1}$
Observation model

Another example

- **States:** X = {home, office, cafe}
- **Observations:** E = {sms, facebook, email}



Transition Probabilities

	home	office	cafe
home	0.2	0.6	0.2
office	0.5	0.2	0.3
cafe	0.2	0.8	0.0

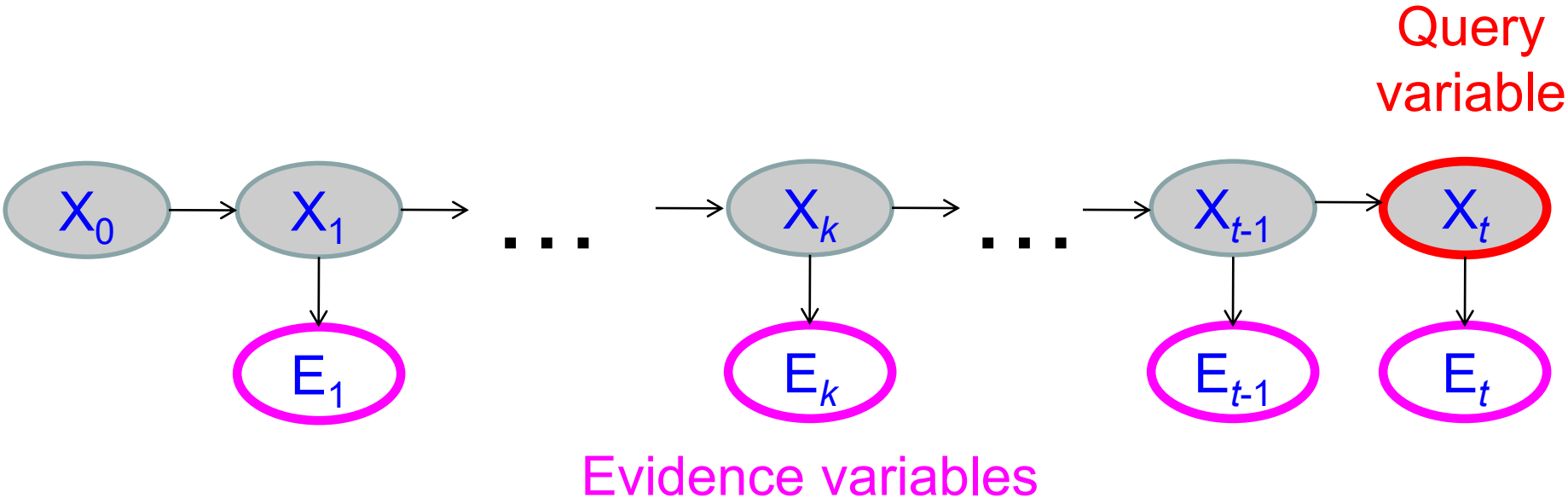
Emission Probabilities

	sms	facebook	email
home	0.3	0.5	0.2
office	0.1	0.1	0.8
cafe	0.8	0.1	0.1

Slide credit: Andy White

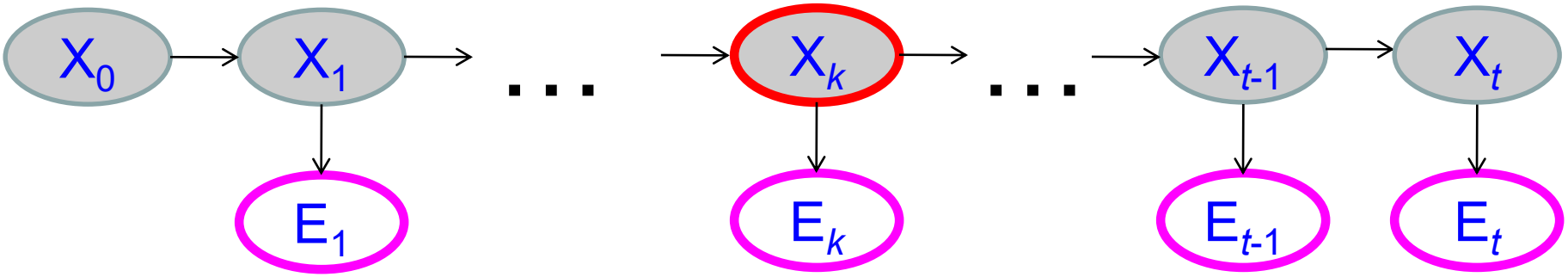
HMM inference tasks

- **Filtering:** what is the distribution over the current state X_t given all the evidence so far, $\mathbf{e}_{1:t}$?
  - The forward algorithm



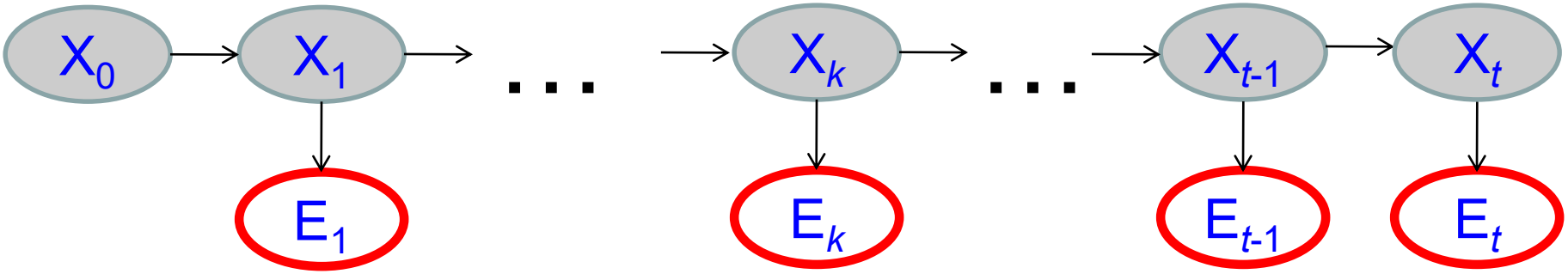
HMM inference tasks

- **Filtering:** what is the distribution over the current state X_t given all the evidence so far, $\mathbf{e}_{1:t}$?
- **Smoothing:** what is the distribution of some state X_k given the entire observation sequence $\mathbf{e}_{1:t}$?
  - The forward-backward algorithm



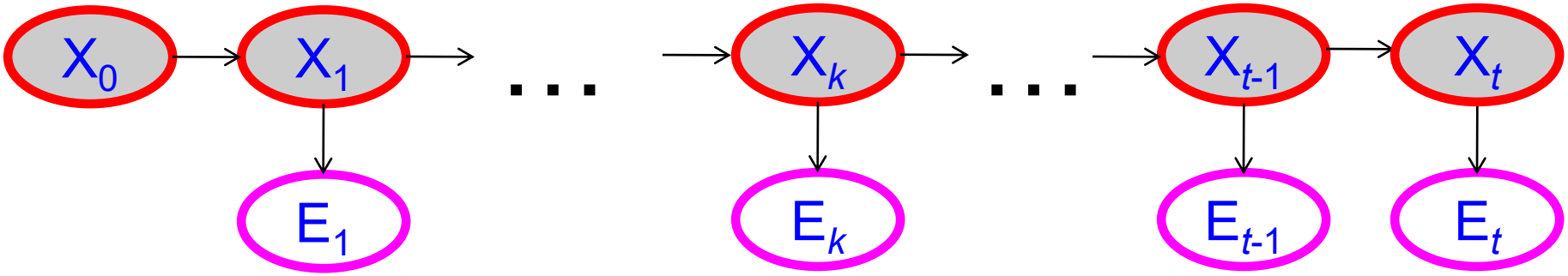
HMM inference tasks

- **Filtering:** what is the distribution over the current state X_t given all the evidence so far, $\mathbf{e}_{1:t}$?
- **Smoothing:** what is the distribution of some state X_k given the entire observation sequence $\mathbf{e}_{1:t}$?
- **Evaluation:** compute the probability of a given observation sequence $\mathbf{e}_{1:t}$



HMM inference tasks

- **Filtering:** what is the distribution over the current state X_t given all the evidence so far, $\mathbf{e}_{1:t}$
- **Smoothing:** what is the distribution of some state X_k given the entire observation sequence $\mathbf{e}_{1:t}$?
- **Evaluation:** compute the probability of a given observation sequence $\mathbf{e}_{1:t}$
- **Decoding:** what is the most likely state sequence $\mathbf{X}_{0:t}$ given the observation sequence $\mathbf{e}_{1:t}$?
  - The Viterbi algorithm



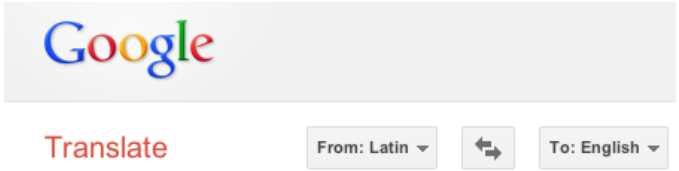
HMM Learning and Inference

- Inference tasks
  - **Filtering:** what is the distribution over the current state X_t given all the evidence so far, $\mathbf{e}_{1:t}$
  - **Smoothing:** what is the distribution of some state X_k given the entire observation sequence $\mathbf{e}_{1:t}$?
  - **Evaluation:** compute the probability of a given observation sequence $\mathbf{e}_{1:t}$
  - **Decoding:** what is the most likely state sequence $\mathbf{X}_{0:t}$ given the observation sequence $\mathbf{e}_{1:t}$?
- Learning
  - Given a training sample of sequences, learn the model parameters (transition and emission probabilities)
    - EM algorithm

Applications of HMMs

- Speech recognition HMMs:
  - Observations are acoustic signals (continuous valued)
  - States are specific positions in specific words (so, tens of thousands)

- Machine translation HMMs:
  - Observations are words (tens of thousands)
  - States are translation options

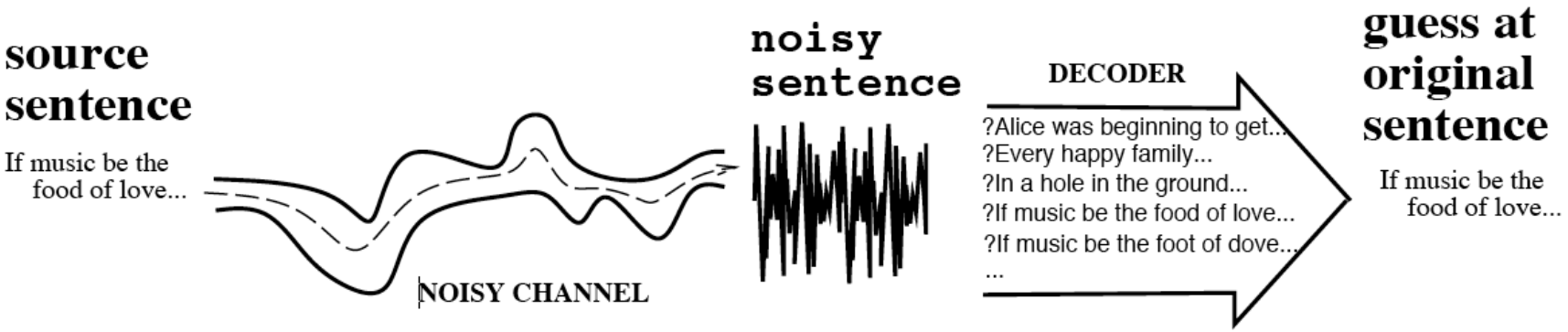


- Robot tracking:
  - Observations are range readings (continuous)
  - States are positions on a map (continuous)

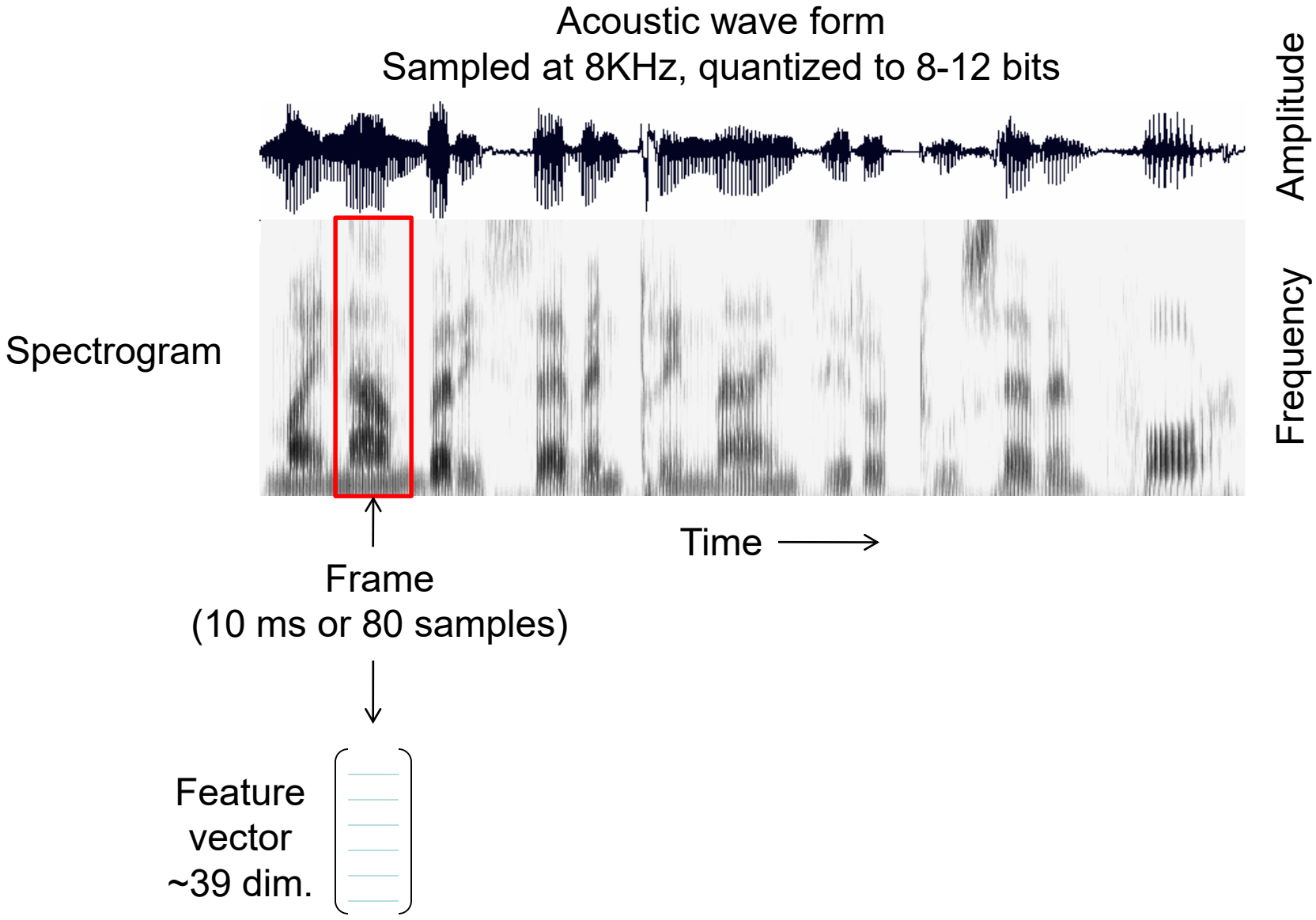
Source: Tamara Berg

Application of HMMs: Speech recognition

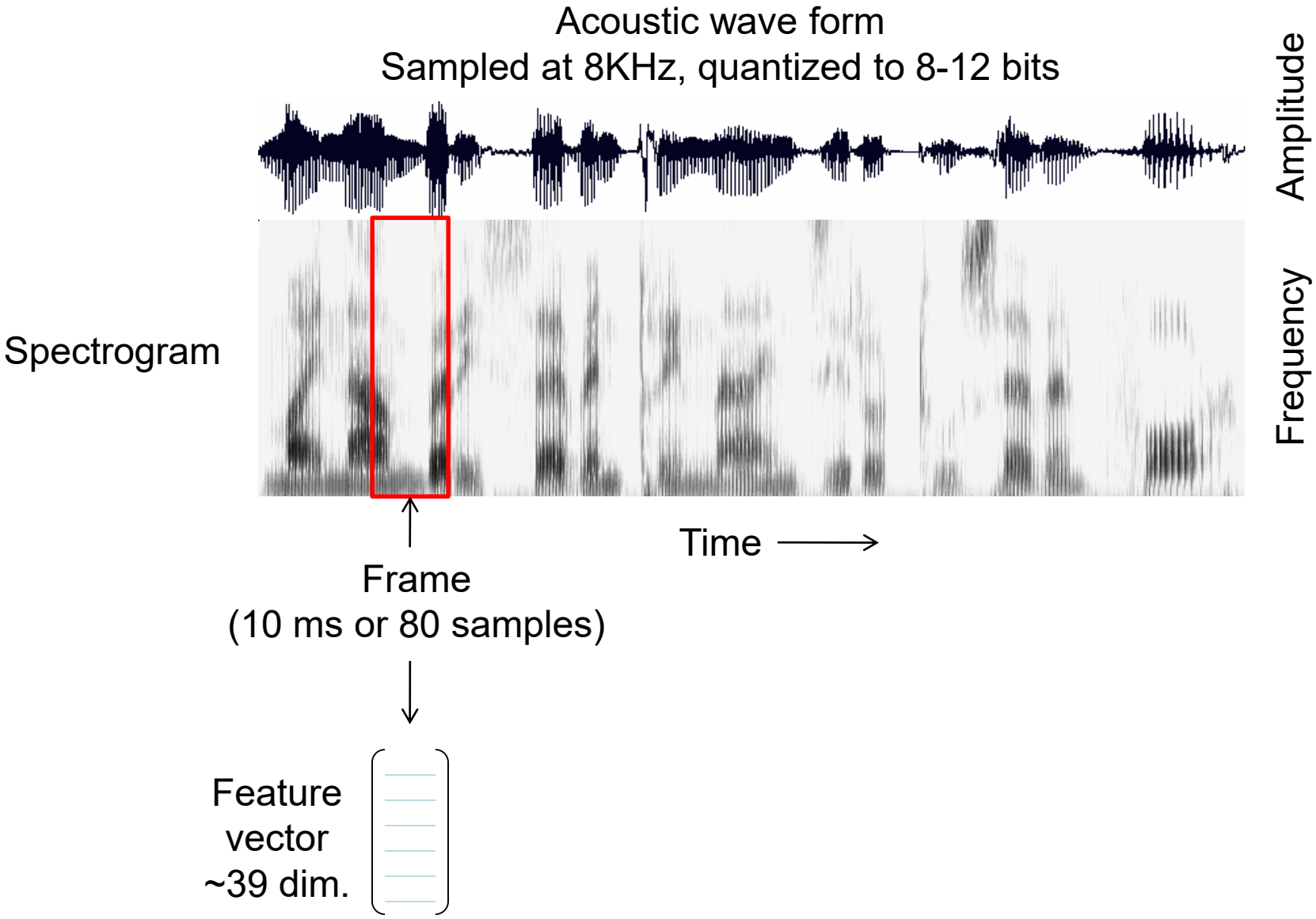
- "Noisy channel" model of speech



Speech feature extraction



Speech feature extraction



Phonetic model

- **Phones:** speech sounds
- **Phonemes:** groups of speech sounds that have a unique meaning/function in a language (e.g., there are several different ways to pronounce “t”)

Phonetic model

IPA Symbol	ARPAbet Symbol	Word	IPA Transcription	ARPAbet Transcription
[p]	[p]	parsley	['parsli]	[p aa r s l iy]
[t]	[t]	tarragon	['tærəgɑn]	[t ae r ax g aa n]
[k]	[k]	catnip	['kætnɪp]	[k ae t n ix p]
[b]	[b]	bay	[beɪ]	[b ey]
[d]	[d]	dill	[dɪl]	[d ih l]
[g]	[g]	garlic	['gɑrlɪk]	[g aa r l ix k]
[m]	[m]	mint	[mɪnt]	[m ih n t]
[n]	[n]	nutmeg	['nʌtmɛg]	[n ah t m eh g
[ŋ]	[ng]	ginseng	['dʒɪnsɪŋ]	[jh ih n s ix ng]
[f]	[f]	fennel	['fɛnl̩]	[f eh n el]
[v]	[v]	clove	[kloʊv]	[k l ow v]
[θ]	[th]	thistle	['θɪsl̩]	[th ih s el]
[ð]	[dh]	heather	['hɛðɚ]	[h eh dh axr]
[s]	[s]	sage	[seɪdʒ]	[s ey jh]
[z]	[z]	hazelnut	['heɪzl̩nʌt]	[h ey z el n ah t]
[ʃ]	[sh]	squash	[skwɑʃ]	[s k w a sh]
[ʒ]	[zh]	ambrosia	[æm'broʊʒə]	[ae m b r ow zh ax]
[tʃ]	[ch]	chicory	['tʃɪkəɹi]	[ch ih k axr iy]
[dʒ]	[jh]	sage	[seɪdʒ]	[s ey jh]
[l]	[l]	licorice	['lɪkəɹɪʃ]	[l ih k axr ix sh]
[w]	[w]	kiwi	['kiwi]	[k iy w iy]
[r]	[r]	parsley	['pɑrsli]	[p aa r s l iy]
[j]	[y]	yew	[yu]	[y uw]
[h]	[h]	horseradish	['hɔrsrædɪʃ]	[h ao r s r ae d ih sh]
[ʔ]	[q]	uh-oh	[ʔʌʔoʊ]	[q ah q ow]
[ɾ]	[dx]	butter	['bʌɾɚ]	[b ah dx axr]
[ɾ̃]	[nx]	wintergreen	[wɪɾ̃əgrin]	[w ih nx axr g r i n]
[l̩]	[el]	thistle	['θɪsl̩]	[th ih s el]

Figure 4.1 IPA and ARPAbet symbols for transcription of English consonants.

IPA Symbol	ARPAbet Symbol	Word	IPA Transcription	ARPAbet Transcription
[i]	[iy]	lily	['lɪli]	[l ih l iy]
[ɪ]	[ih]	lily	['lɪli]	[l ih l iy]
[eɪ]	[ey]	daisy	['deɪzi]	[d ey z i]
[ɛ]	[eh]	poinsettia	[pɔɪn'sɛriə]	[p oy n s eh dx iy ax]
[æ]	[ae]	aster	['æstɚ]	[ae s t axr]
[ɑ]	[aa]	poppy	['papi]	[p aa p i]
[ɔ]	[ao]	orchid	['ɔrkɪd]	[ao r k ix d]
[ʊ]	[uh]	woodruff	['wʊdrʌf]	[w uh d r ah f]
[oʊ]	[ow]	lotus	['loʊrəs]	[l ow dx ax s]
[u]	[uw]	tulip	['tulɪp]	[t uw l ix p]
[ʌ]	[uh]	buttercup	['bʌɾɚ,kʌp]	[b uh dx axr k uh p]
[ɝ]	[er]	bird	['bɝd]	[b er d]
[aɪ]	[ay]	iris	['aɪɹɪs]	[ay r ix s]
[aʊ]	[aw]	sunflower	['sʌnflaʊɚ]	[s ah n f l aw axr]
[ɔɪ]	[oy]	poinsettia	[pɔɪn'sɛriə]	[p oy n s eh dx iy ax]
[ju]	[y uw]	feverfew	[fivɚfju]	[f iy v axr f y u]
[ə]	[ax]	woodruff	['wʊdrəf]	[w uh d r ax f]
[ɚ]	[axr]	heather	['hɛðɚ]	[h eh dh axr]
[ɨ]	[ix]	tulip	['tulɪp]	[t uw l ix p]
[ʉ]	[ux]		[]	[]

Figure 4.2 IPA and ARPAbet symbols for transcription of English vowels

HMM models for phones

- HMM states in most speech recognition systems correspond to *subphones*
  - There are around 60 phones and as many as 60^3 context-dependent *triphones*

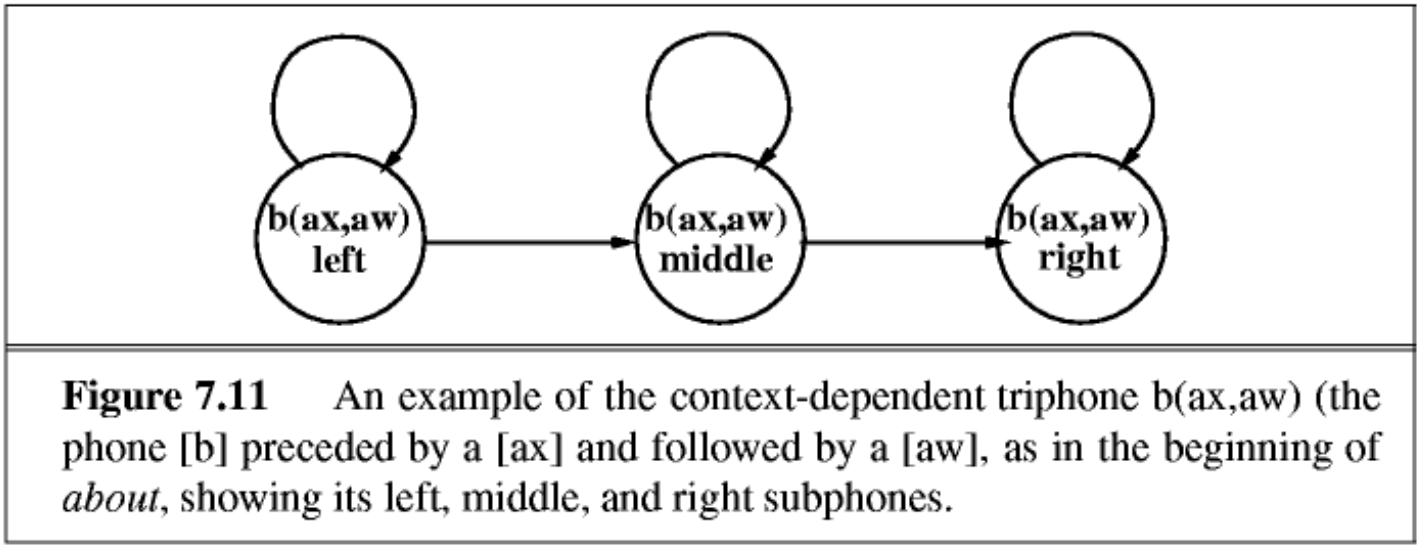


Figure 7.11 An example of the context-dependent triphone b(ax,aw) (the phone [b] preceded by a [ax] and followed by a [aw], as in the beginning of *about*, showing its left, middle, and right subphones.

HMM models for words

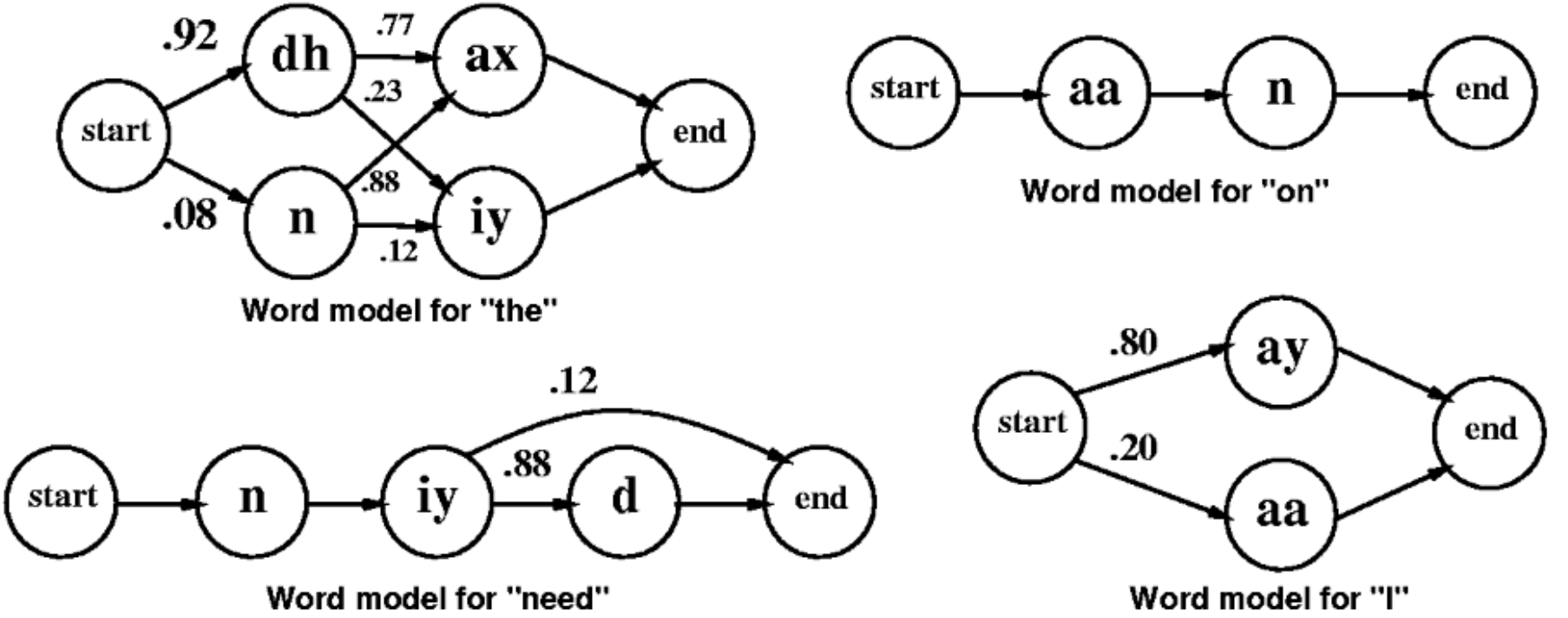


Figure 7.5 Pronunciation networks for the words *I*, *on*, *need*, and *the*. All networks (especially *the*) are significantly simplified.