Supervised learning: a summary

- A supervised learning paradigm assumes that there are correct labels, sequences of labels, or trees and graphs.
- Having correct labels allows us to compare th model with the correct labels to compute the of a model.
- During traini d use these initi e how wrong th d update the parameters to reduce this loss.
- When the training is done, we make predic for the label with the highest score.
- The key to supervised learning is to have annotated data with correct labels. Is there anything we can do without annotated data?

Beyond Supervised Learning

There are other learning scenarios where labeled training sets are available to various degree or not available at all

- When there is no labeled data at all, we'll have to d **unsupervis** ring, variants of the EM algori
- When there is a small amount of labeled dat to try **semi-supervised learning**
- When there is a lot of labeled data in one doma only a small of labeled data in the target domain, we might try **domain adaptation**

K-Means clustering algorithm

K-means clustering algorithm

procedure K-MEANS($\boldsymbol{x}_{1:N}, K$)

  for $i \in 1 \cdots N$ **do**

    $z^{(i)} \leftarrow$ RANDOMINT$(1, K)$ ▷ I ... mbership

  repeat

    for k ... ute cluster centers

      $\boldsymbol{v}_k \leftarrow \frac{1}{\sum_{i=1}^{N} \delta(z^{(i)}=k)} \sum_i^N \ldots i)$

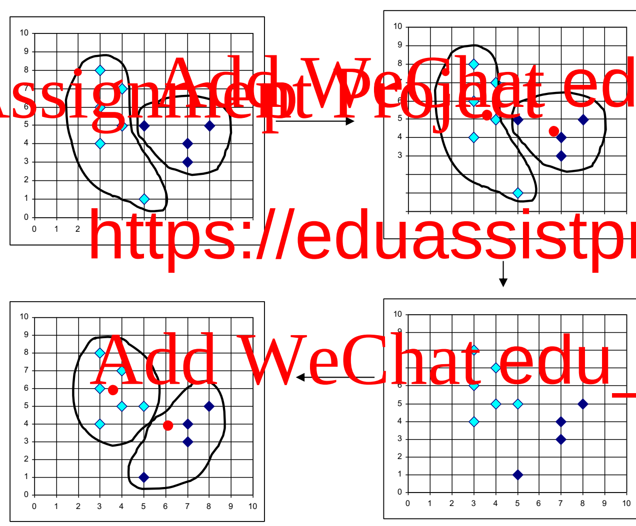
    for $i \in 1 \cdots N$ **do**

      $z^{(i)} \leftarrow \text{argmin}_K \, ||\boldsymbol{x}^{(i)} - \boldsymbol{v}_k||^2$

  until Converged

  return $z^{(i)}$

K-Means training



- K-means clustering is non-parametric and has no parameters to update
- As a result, there is also no separate training and test phase.
- The number of clusters needs to be pre-specified before the

Semi-supervised learning

- Initialize parameters with supervised learning and then apply unsupervised learning (such as the EM algorithm)
- Multi-view learning: Co-training

  - divide features into multiple views, and tra each vie
  - Each cla instances, using only the features ava predictions are then used as ground tru classifiers associated with the other vi

  - Named entity example: named entity view and local context view
  - Word sense disambiguation: local context view and global context view

Multi-view Learning: co-training

https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Domain adaptation

Supervised domain ad[illegible] n adaptation (Daumé III, 2007)

- Creates copies of each feature: one for each domain and one for the cross-domain setting

$$\boldsymbol{f}(\boldsymbol{x}, y, d) = \{(\text{boring, NEG, Movie}):1,\ (\text{bor}[illegible]$$
$$(\text{fl}[illegible]):1,$$
$$(\text{t}[illegible]\text{ld, NEG, *}):1,$$
$$\ldots\}$$

where d is the domain.

- Let the learning algorithm allocate weights between domain specific features and cross-domain features: for words that facilitate prediction in both domains, the learner will use cross-domain features. For words that are only relevant to a particular domain, domain-specific features will be used.

Other learning paradigms

- **Active learning:** A learning that is often used to reduce the number of instances that have to be annotate produce the same level of accuracy
- **Distant sup** ou can generate so e external resource such as a dictionary. Fo generate named entity annotation wit
- **Multitask learning:** The learnin tation that can be used to solve multiple tasks (learning POS tagging with syntactic parsing)

Expectation Maximization

- An unsupervised iterative learning procedure that has two steps: the Expectation Step and the Maximiz

- Has many applications in NLP: POS tagging parsing, wor
  - The mos t step in sta

- Efficient incarnations with dynamic pr

  - The Forward-Backward algorith
  - The Inside-Outside algorithm (for parsing)
- The main workhorse for unsupervised learning in NLP

Expectation Maximization for Sequence Labeling

- The naïve E
- A more efficie

lgorithm

Hidden Markov assumptions

- Recall the genera

$$P(\boldsymbol{x},\boldsymbol{y}) = P(\boldsymbol{x}|\boldsymbol{y})P(\boldsymbol{y})$$

- Apply the conditional independence assu only depends on its corresponding tag:

- Apply the assumption that a tag only depends on its previous tag:

$$P(\boldsymbol{x},\boldsymbol{y}) = \prod_{m=1}^{M} P(x_m|y_m)P(y_m|y_{m-1})$$

Parameter estimation

- In a supervised setting, the probabilities of the HMM parameters can be computed via a simple count.

- Given two labeled examples:
  - John_N likes_V apples_N

  - John_N likes_V oranges_N
- Counts for the [illegible] tions:

	[illegible]	[illegible]	[illegible]
w_0=John	[illegible]	[illegible]	[illegible]
w_0=likes	[illegible]	[illegible]	[illegible]
w_0=apples	1	0	0
w_0=oranges	1	0	0
$t_{-1} = \Diamond$	2	0	0
t_{-1}=N	0	2	2
t_{-1}=V	2	0	0

Parameter estimation

- Given these count … es of the parameters

	counts			probabilities		
	N	V	♦	N	V	♦
w_0=John	2	0	0			
w_0=likes	0	2	0			
w_0=apples	1	0	0			
						0
t …						0
t_{-1}=N	0	2				0.5
t_{-1}=V	2	0				0

- We implicitly assume the probabilities … re zero:

$$P(\text{John\_V, likes\_N, apples\_V}) = 0$$
$$P(\text{John\_N, likes\_N, apples\_N}) = 0$$
$$\ldots\ldots$$

What if we don't have labeled data?

- Now assume we do
  word tokens:
  - John likes apples
  - John likes oranges
- We can no longer count the frequencies of bigr
  sequences and how often a tag occurs with a wor
- Instead, let's s
  parameters

	Expected co			bilities		
	N	V	♦			♦
w_0=John	-	-	-	0.5	0.1	0
w_0=likes	-	-	-	0.1	0.5	0
w_0=apples	-	-	-	0.2	0.2	0
w_0=oranges	-	-	-	0.2	0.2	0
$t_{-1} = \Diamond$	-	-	-	0.8	0.2	0
t_{-1}=N	-	-	-	0.1	0.6	0.3
t_{-1}=V	-	-	-	0.6	0.2	0.2

Assignment Project
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Joint probability of a token sequence and its tag sequence

With the initial assignm
of the token sequence an

$P(\text{John\_N, likes\_N, apples\_N}) = P(N|\Diamond)P(N|N)P(N|N)P(\blacklozenge|N)P(John|N)$
$\quad P(likes|N)P(apples|N) = 0.8 \times 0.1 \times 0.1 \times 0.3 \times 0.5 \times 0.1 \times 0.2 = 0.000024$

$P(\text{John\_N, likes\_N, apples\_V}) = 0.000096$

$P(\text{John\_N, likes\_V, apples\_V}) = 0.00096$

$P(\text{John\_N, likes\_V, apples\_N}) = 0.00432$

$P(\text{John\_V, likes\_}$

$P(\text{John\_V, likes\_}$

$P(\text{John\_V, likes\_V, apples\_N}) = 0.000072$

$P(\text{John\_V, likes\_V, apples\_V}) = 0.000016$

$P(\text{John\_N, likes\_N, oranges\_N}) = 0.000024$

$P(\text{John\_N, likes\_N, oranges\_V}) = 0.000096$

$P(\text{John\_N, likes\_V, oranges\_V}) = 0.00096$

$P(\text{John\_N, likes\_V, oranges\_N}) = 0.00432$

$P(\text{John\_V, likes\_N, oranges\_N}) = 0.0000072$

$P(\text{John\_V, likes\_N, oranges\_V}) = 0.000029$

$P(\text{John\_V, likes\_V, oranges\_N}) = 0.000072$

$P(\text{John\_V, likes\_V, oranges\_V}) = 0.000016$

Conditional probability of a tag sequence given its token sequence

With the joint probabili
probability of a tag sequence given its token sequence

$P((N, N, N)|(\text{John, likes, apples})) = 0.0043$

$P((N, N, V)|(\text{John, likes, apples}))$

$P((N, V, V)|(\text{John, likes, apples}))$

$P((N, V, N)|(\text{John, likes, apples}))$

3

2

$P((V, V, N)|(\text{John, likes, app}$

$P((V, V, V)|(\text{John, likes, app}$

$P((N, N, N)|(\text{John, likes, ora}$

$P((N, N, V)|(\text{John, likes, oranges})) = 0.0174$

$P((N, V, V)|(\text{John, likes, oranges})) = 0.174$

$P((N, V, N)|(\text{John, likes, oranges})) = 0.782$

$P((V, N, N)|(\text{John, likes, oranges})) = 0.0013$

$P((V, N, V)|(\text{John, likes, oranges})) = 0.0052$

$P((V, V, N)|(\text{John, likes, oranges})) = 0.013$

$P((V, V, V)|(\text{John, likes, oranges})) = 0.0029$

Expected counts of the parameters

With the conditional probabilities of each tag sequence given its token sequence, we can compute the expected count of each parameter – the count of each parameter weighted by the conditional probability of the tagged sequence it a

						bilities
						$\blacklozenge$
w_0=John	1.955	0.0448	0			-
w_0=likes	0.056	?	0			-
w_0=apples	0.80	?	0	-	-	-
w_0=oranges	0.80	?	?	-	-	-
$t_{-1} = \Diamond$	1.995	0.0448	0	-	-	-
t_{-1}=N	0.0546	1.957	1.601	-	-	-
t_{-1}=V	?	?	?	-	-	-

Maximization

With the expected coun
maxization) the parameters, and replace the initial parameters with the updated parameters:

	Expected counts			lities		
					V	♦
w_0=John				4	-	-
w_0=likes	0.056	?	0		-	-
w_0=apples	0.80	?	0		-	-
w_0=oranges	0.80	?	?	-	-	-
$t_{-1} = \Diamond$	1.995	0.0448	0	0.978	-	-
t_{-1}=N	0.0546	1.957	1.601	-	0.5417	-
t_{-1}=V	?	?	?	-	-	-

With the updated parameters, we can iterate this process...

A summary of this process:

- We first initialize the parameters with some initial values, hopefully with some prior knowledge

- The E-Step:
  - Using these parameters, we can compute t probab
  - With the expect

- The M-step: we then re-estimate the par maximizing them.

- We repeat this process until it converges at some local maxima. The underlying model is not concave (the opposite of convex) so there is no guarantee that it will hit global maxima.

The generic EM algorithm

Input: A sample of N points, $x^{(1)}, x^{(2)}, \cdots, x^{(N)}$. A model $P(x, y|\boldsymbol{\theta})$ of the following form: $P(x, y|\boldsymbol{\theta}) = \prod_{r=1}^{|\theta|} \boldsymbol{\theta}_r^{Count(x,y,r)}$

Output: $\boldsymbol{\theta}^T$

Initialization: Choose some initial value for

for $t \leftarrow 1 \cdots T$ **do**
  for $r = 1$
    $\mathbb{E}[Cou$

  for $i = 1 \cdots N$ **do**
    For all y, compute $t_y = P(x^i$

    for all y, set $u_y = t_y / \sum_y t_y$
    for all $r = 1 \cdots |\boldsymbol{\theta}|$, set $\mathbb{E}[Count(r)] = \mathbb{E}[Count(r)] + \sum_y u_y Count(x^i, y, r)$
for $r = 1 \cdots |\boldsymbol{\theta}|$ **do**
  $\boldsymbol{\theta}_r^t = \frac{\mathbb{E}[Count(r)]}{Z}$ where Z is a normalization constant

At this time you should have some questions...

- Does this work? Why does this work at all?
- What are the scenarios in which EM can be depl

- Can this be don
- Why do we wei

  sequence (and not something else)?
- Will the log likelihood monotonically inc

  iteration?
- We'll try to answer some of them...

The Baum-Welch algorithm

- The naive Expectation Maximization algorithm we outlined above works for short sentences in small data sets, but does not scale.
- The Baum-Welch algorithm is an efficient alte combines E
- In the M-step, t expected count:

$$Pr(W = i|Y = k) = \phi_{k,i} = \frac{\mathbb{E}[\text{co} \quad k)]}{\mathbb{E}[\text{count}(Y = k)]}$$

$$Pr(Y_m = k|Y_{m-1} = k') = \lambda_{k',k} = \frac{\mathbb{E}[\text{count}(Y_m = k, Y_{m-1} = k')]}{\mathbb{E}[\text{count}(Y_{m-1} = k')]}$$

The E-Step: transition counts

- The local scores follow the usual definitions of HMM:

$$s_m(k, k') = \log P_E(w_m | Y_m = k; \phi) + \log \quad {}_{m-1} = k'; \lambda)$$

- The expected transition count for a single inst

$$\mathbb{E}[\text{count}(Y_m \quad = k, Y_{m-1} = k' | \boldsymbol{w})$$

$$Pr(Y_m = k, Y_{m-1} = k' | \boldsymbol{w}) = \frac{\alpha_m \quad ; k') \times \beta_m(k)}{\quad {}_{M+1} \quad)}$$

- The posterior is computed the same way as in the forward-backward computation in CRF, with the only difference being how the local score is computed.

The E-Step: emission counts

- The local scores for

$$s_m(k, k') = \log P_E(w_m|Y_m = k; \phi) + \log P_T(Y_m = k|Y_{m-1} = k'; \lambda)$$

- The expected emission count for a single insta

$$\mathbb{E}[\text{co} \quad \ldots \quad {}_m = k|\boldsymbol{w})$$

$$Pr(\;{}_m \quad \ldots \quad {}_m \quad {}_m \;{}_1 = k'|\boldsymbol{w})$$

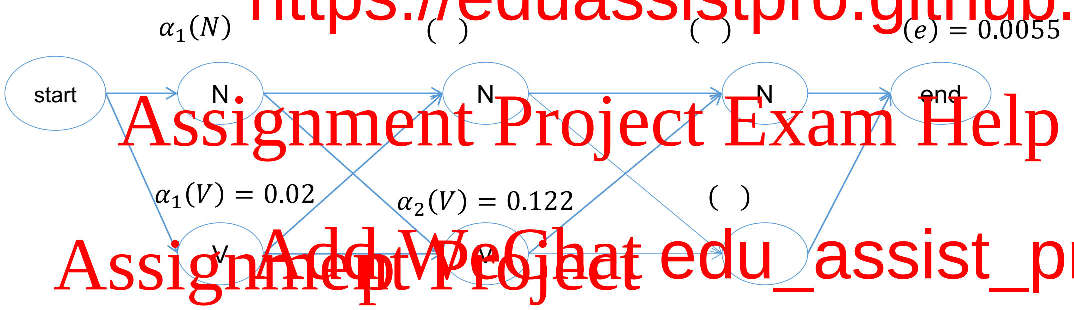
$$Y_{m-1}=k'$$

$$= \sum_{Y_{m-1}=k'} \frac{\alpha_{m-1}(k') \times \exp \quad \ldots)}{\alpha_{M+1}(\blacklozenge)}$$

$$= \alpha_m(k)\beta_m(k)$$

- The posterior is computed the same way as in the forward-backward computation in CRF, with the only difference being how the local score is computed.

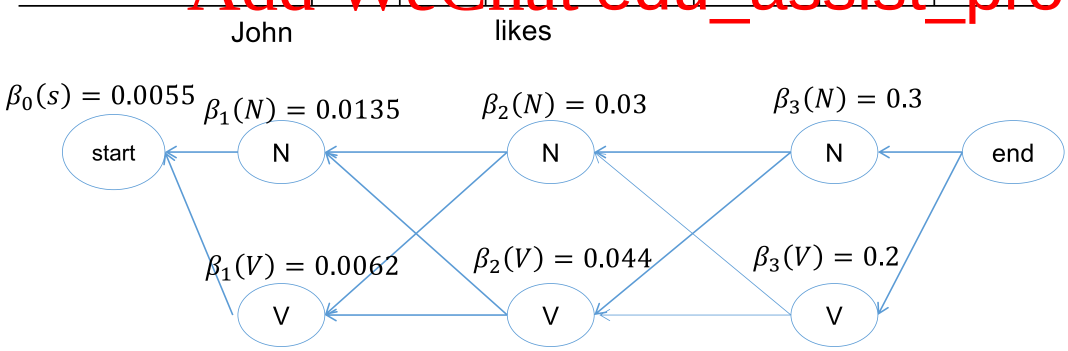
Baum-Welch: Forward computation



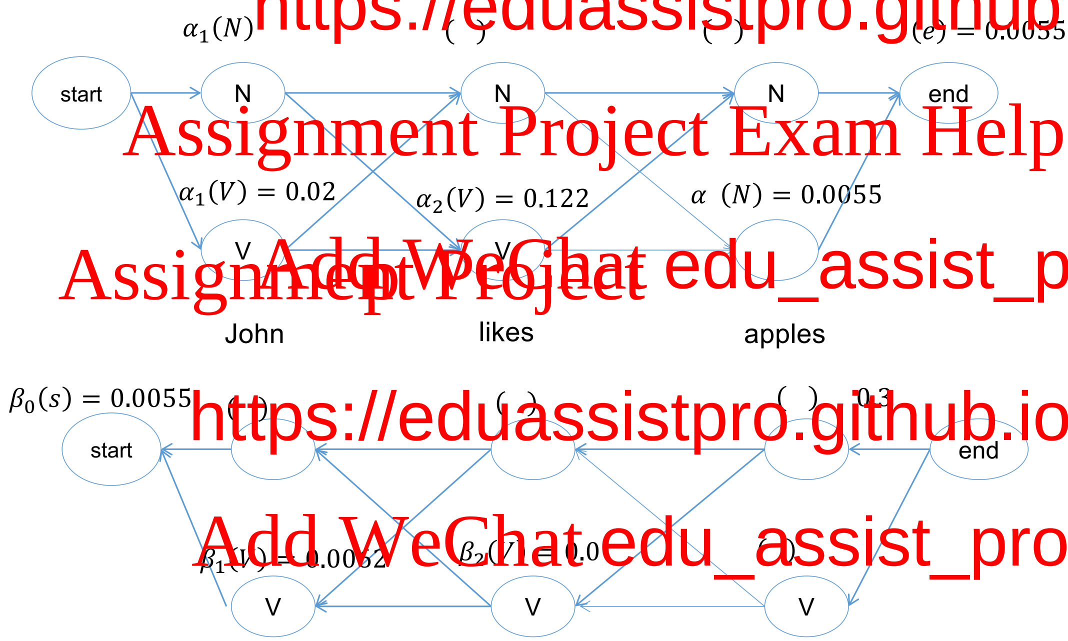
				abilities		
	N	V				♦
w_0=John	-	-	-			0
w_0=likes	-	-	-			0
w_0=apples	-	-	-	0.2	0.2	0
w_0=oranges	-	-	-	0.2	0.2	0
$t_{-1} = \Diamond$	-	-	-	0.8	0.2	0
t_{-1}=N	-	-	-	0.1	0.6	0.3
t_{-1}=V	-	-	-	0.6	0.2	0.2

Baum-Welch: Backward computation

					lities	
	N	V	♦	N	V	♦
w_0=John	-	-	-	0.5	0.1	0
w_0=likes	-	-	-			0
w_0=apples	-	-				0
w_0=or					0.2	0
t_{-1} =					0.2	0
t_{-1}=N						0.3
t_{-1}=V	-	-	-			0.2

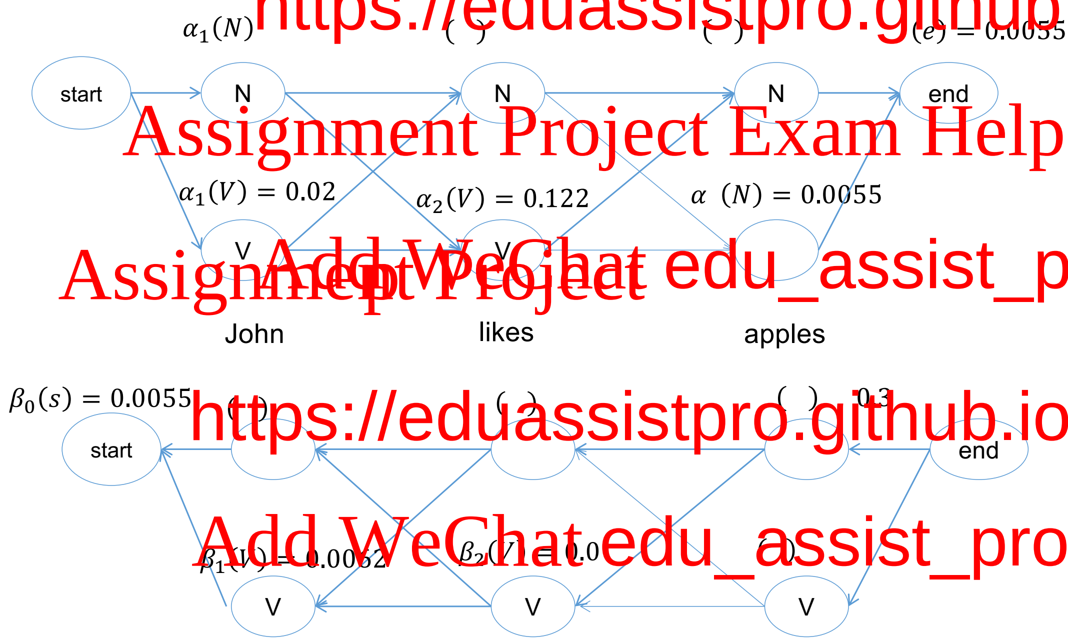


Collecting expected counts



$$\mathbb{E}[count(Y_2 = V, likes)|\text{John likes apples}]$$
$$= \alpha_2(V)\beta_2(V)/\alpha_4(end) = ?$$

Collecting expected counts



$$\mathbb{E}[count(Y_2 = V, Y_1 = N)|\text{John likes apples}]$$
$$= \alpha_1(N)s_2(V, N)\beta_2(V)/\alpha_4(end) = ?$$

Sequence labeling summary

- Decoding: the Vit
- Parameter estimation
  - Supervised algorithms:
    - HMM: there is a closed form solution, as
      independence
    - P
      fe es features
      th
    - CRF: Updates the feature weight ds to
      compute expected feature coun es a
      posterior that can be computed e rd
      backward algorithm.
    - LSTM-CRF: Learned feature representation and transition scores via RNNs.
  - Unsupervised algorithms:
    - The Baum-Welch algorithm, which combines expectation maximization and the forward-backward algorithm.