# Semi-supervised Learning

EECS 395/495 Special Topics in Machine Learning: Probabilistic Graphical Models

# Semi-supervised Learning

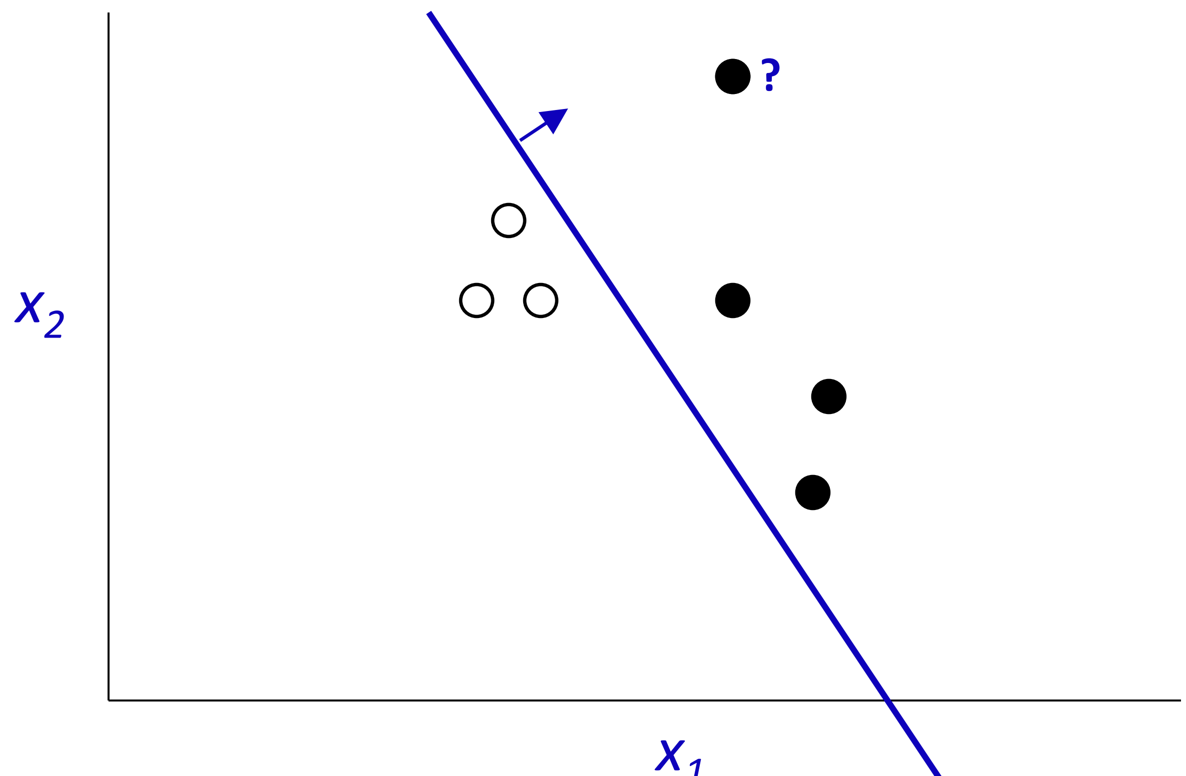
- Unlabeled data abounds in the world
  - Web, measurements, etc.

- *Labeled* data is expensive
  - Image classification, natural language processing, speech recognition, etc. all require large #s of labels

- Idea: use unlabeled data to help with learning

# Supervised Learning

Learn function from $\mathbf{x} = (x_1, \ldots, x_d)$ to $y \in \{0, 1\}$
given labeled examples $(\mathbf{x}, y)$

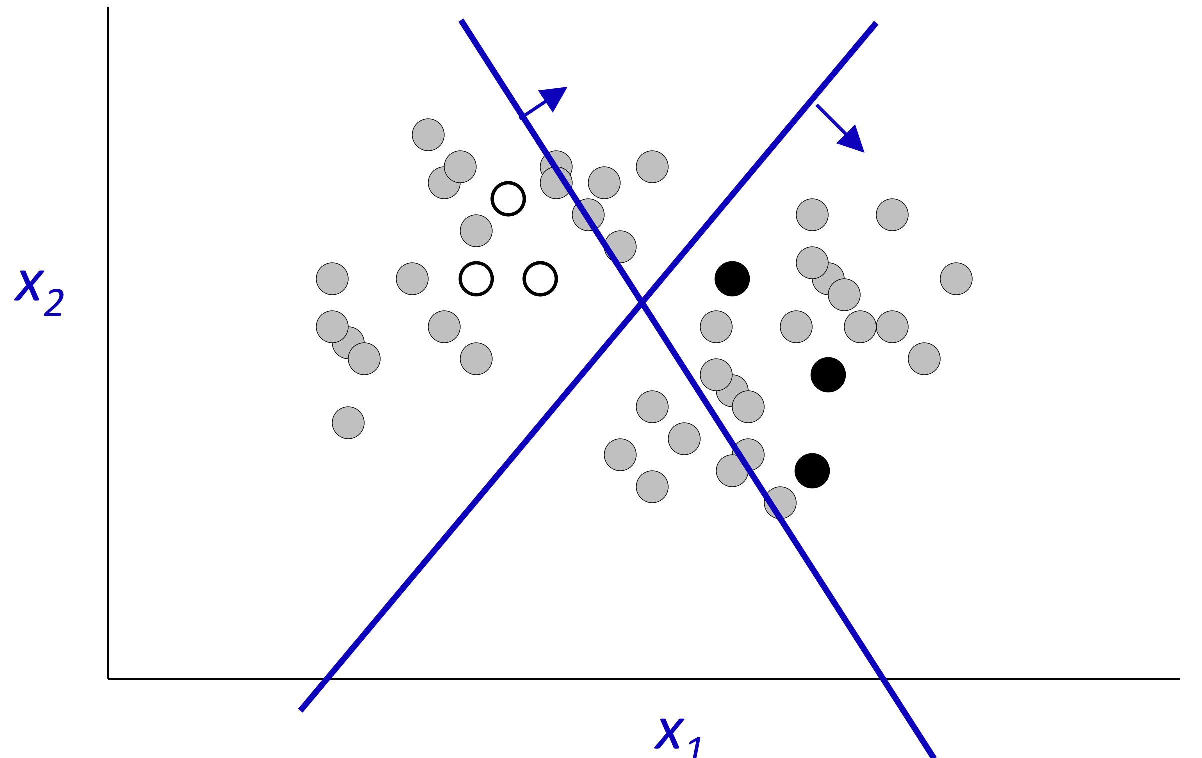


# Semi-Supervised Learning (SSL)

Learn function from $\mathbf{x} = (x_1, \ldots, x_d)$ to $y \in \{0, 1\}$

given labeled examples $(\mathbf{x}, y)$

and unlabeled examples $(\mathbf{x})$



# SSL in Graphical Models

- Graphical Model describes how data (***x***, y) is generated
- Missing Data: *y*
- So use EM

# Example: Document classification with Naïve Bayes

$$P(x_i|\theta) = \sum_{j\in[M]} P(c_j|\theta)P(x_i|c_j;\theta).$$

- $x_i$ = count of word $i$ in document
- $c_j$ = document class (sports, politics, etc.)
- $x_{it}$ = count of word $i$ in docs of class $t$

$$P(x_i|\theta) \propto P(|x_i|) \sum_{j\in[M]} P(c_j|\theta) \prod_{w_t\in\mathcal{X}} P(w_t|c_j;\theta)^{x_{it}}$$

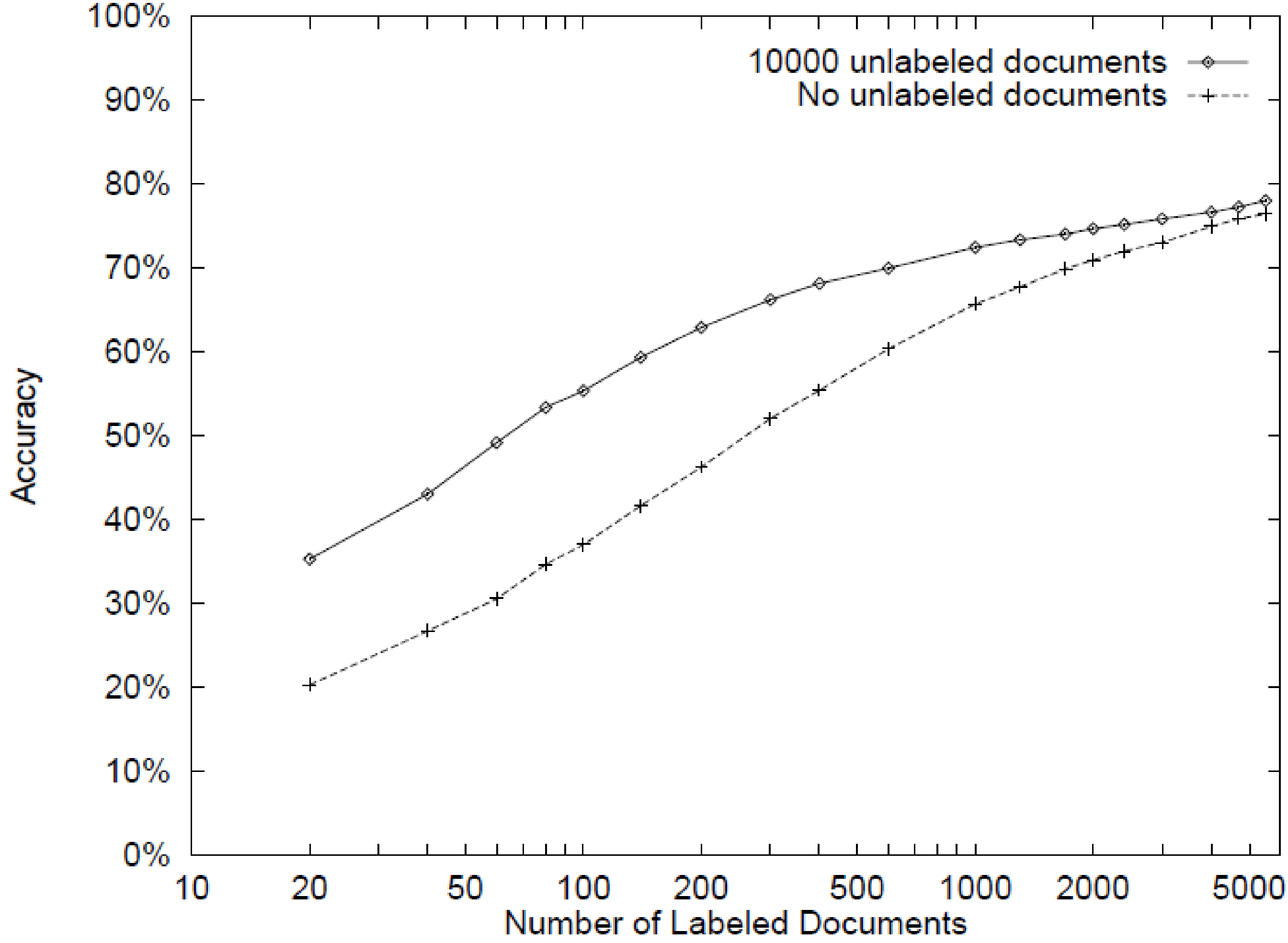
- $M$ classes, $W = |\mathcal{X}|$ words

(from *Semi-supervised Text Classification Using EM*, Nigam, et al.)

# Semi-supervised Training

- Initialize $\theta$ ignoring missing data
- E-step:
  - $E[x_{it}]$ = count of word *i* in docs of class *t* in training set
    + $E_\theta$[count of word *i* in docs of class *t* in unlabeled data]
  - $E[\#c_t]$ = count of docs in class t in training
    + $E_\theta$[count of docs of class *t* in unlabeled data]
- M-step:
  - Set $\theta$ according to expected statistics above, I.e.:
    - $P_\theta(w_t \mid c_t) = (E[x_{it}] + 1) / (W + \Sigma_i E[x_{it}])$
    - $P_\theta(c_t) = (E[\#c_t] + 1) / (\#\text{tokens} + M)$

# Semi-supervised Learning



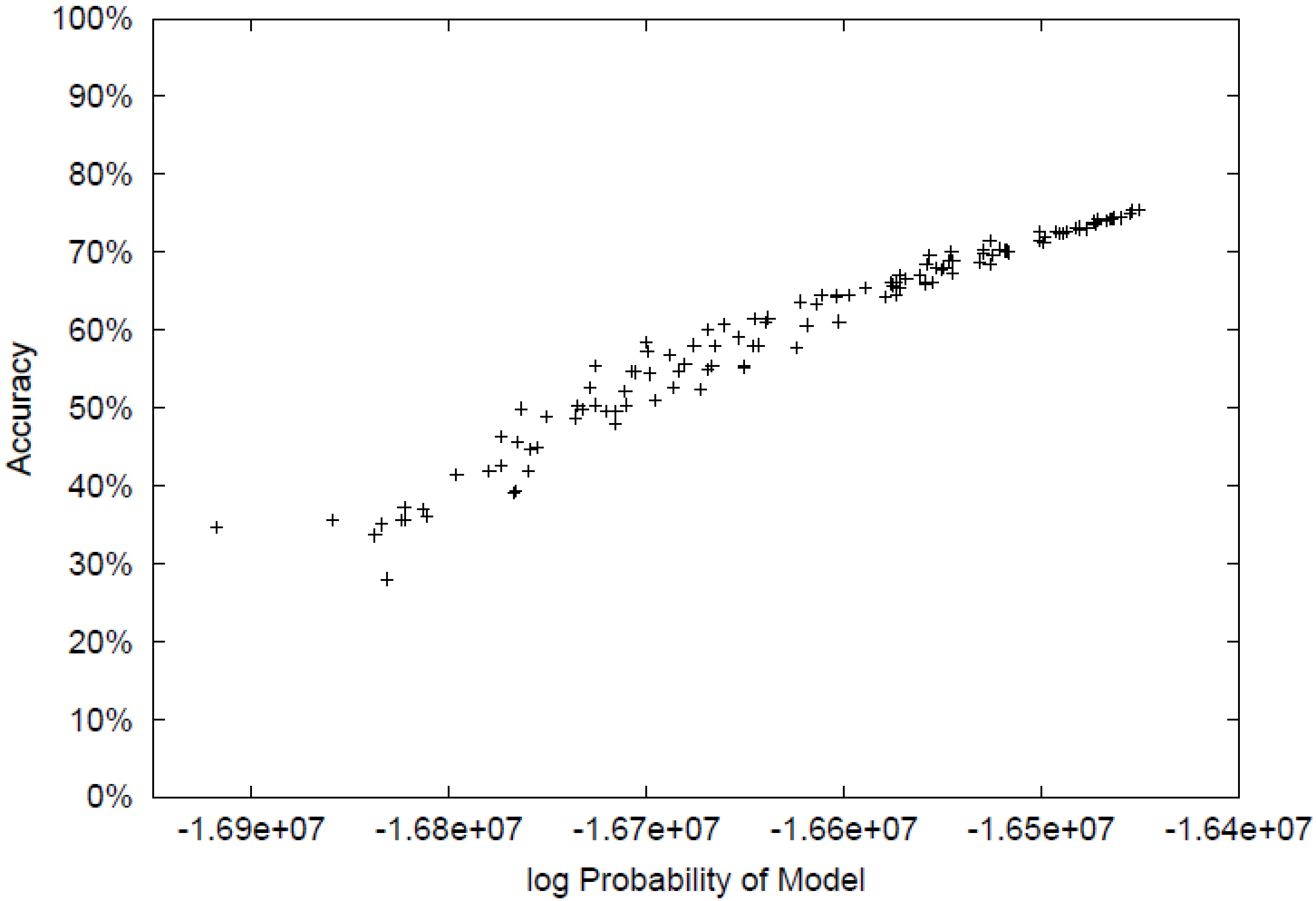
# When does semi-supervised learning work?

- When a better model of $P(\boldsymbol{x})$ -> a better model of $P(y \mid \boldsymbol{x})$

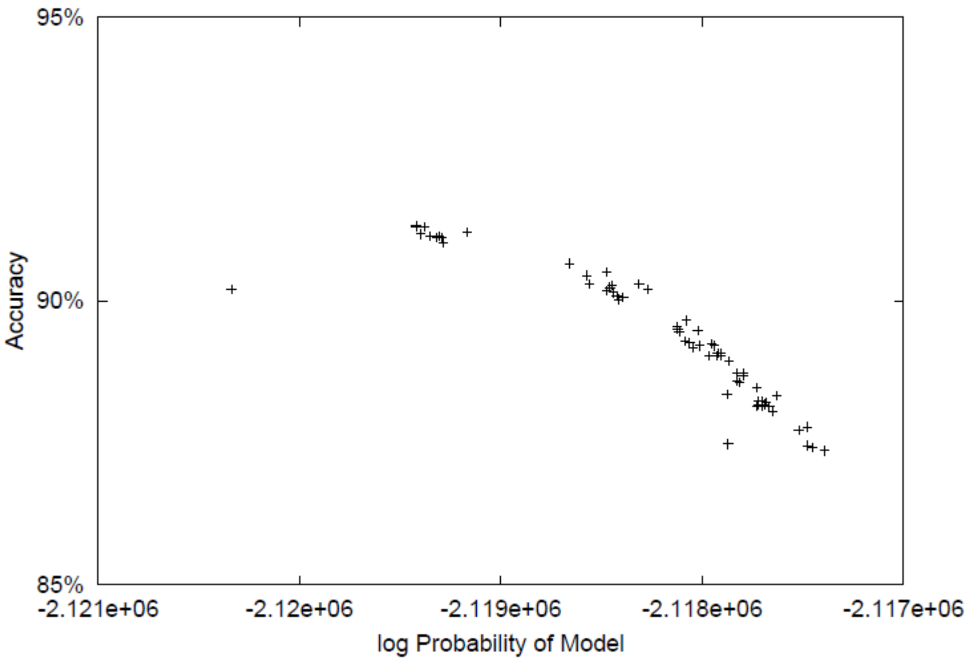
- Can’t use purely *discriminative* models

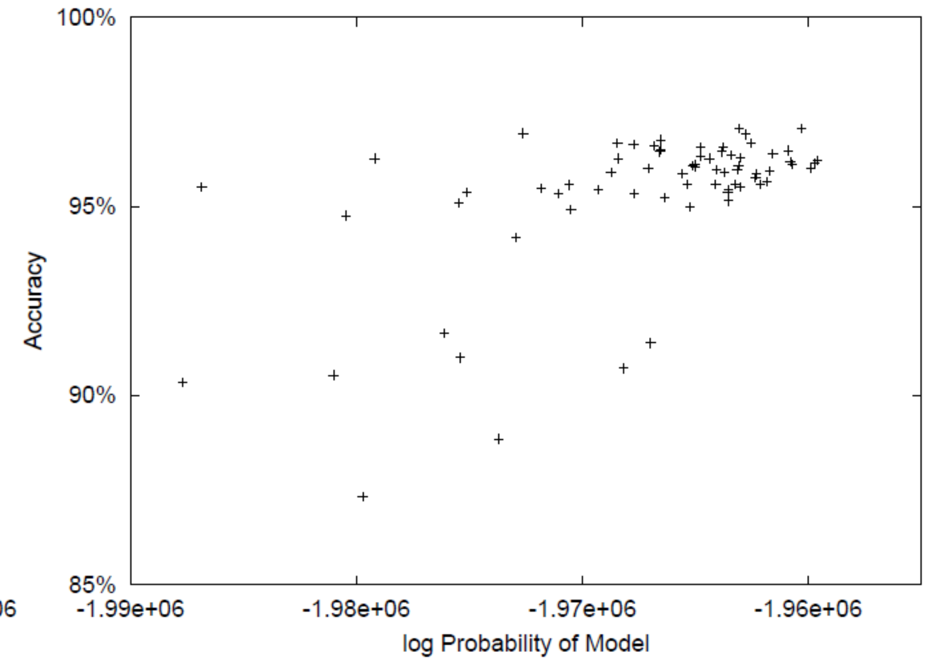
- Accurate modeling assumptions are key
  - Consider: *negative* class

# Good example



# Issue: negative class



# Negative

- NB*, EM* represent the negative class with the optimal number of model classes ($c_i$'s)

| Category | NB1 | EM1 | NB* | EM* |
|---|---|---|---|---|
| acq | 86.9 | 81.3 | 88.0 (4) | **93.1** (10) |
| corn | 94.6 | 93.2 | 96.0 (10) | **97.2** (40) |
| crude | 94.3 | 94.9 | 95.7 (13) | **96.3** (10) |
| earn | 94.9 | 95.2 | **95.9** (5) | 95.7 (10) |
| grain | 94.1 | 93.6 | 96.2 (3) | **96.9** (20) |
| interest | 91.8 | 87.6 | 95.3 (5) | **95.8** (10) |
| money-fx | 93.0 | 90.4 | 94.1 (5) | **95.0** (15) |
| ship | 94.9 | 94.1 | **96.3** (3) | 95.9 (3) |
| trade | 91.8 | 90.2 | 94.3 (5) | **95.0** (20) |
| wheat | 94.0 | 94.5 | 96.2 (4) | **97.8** (40) |

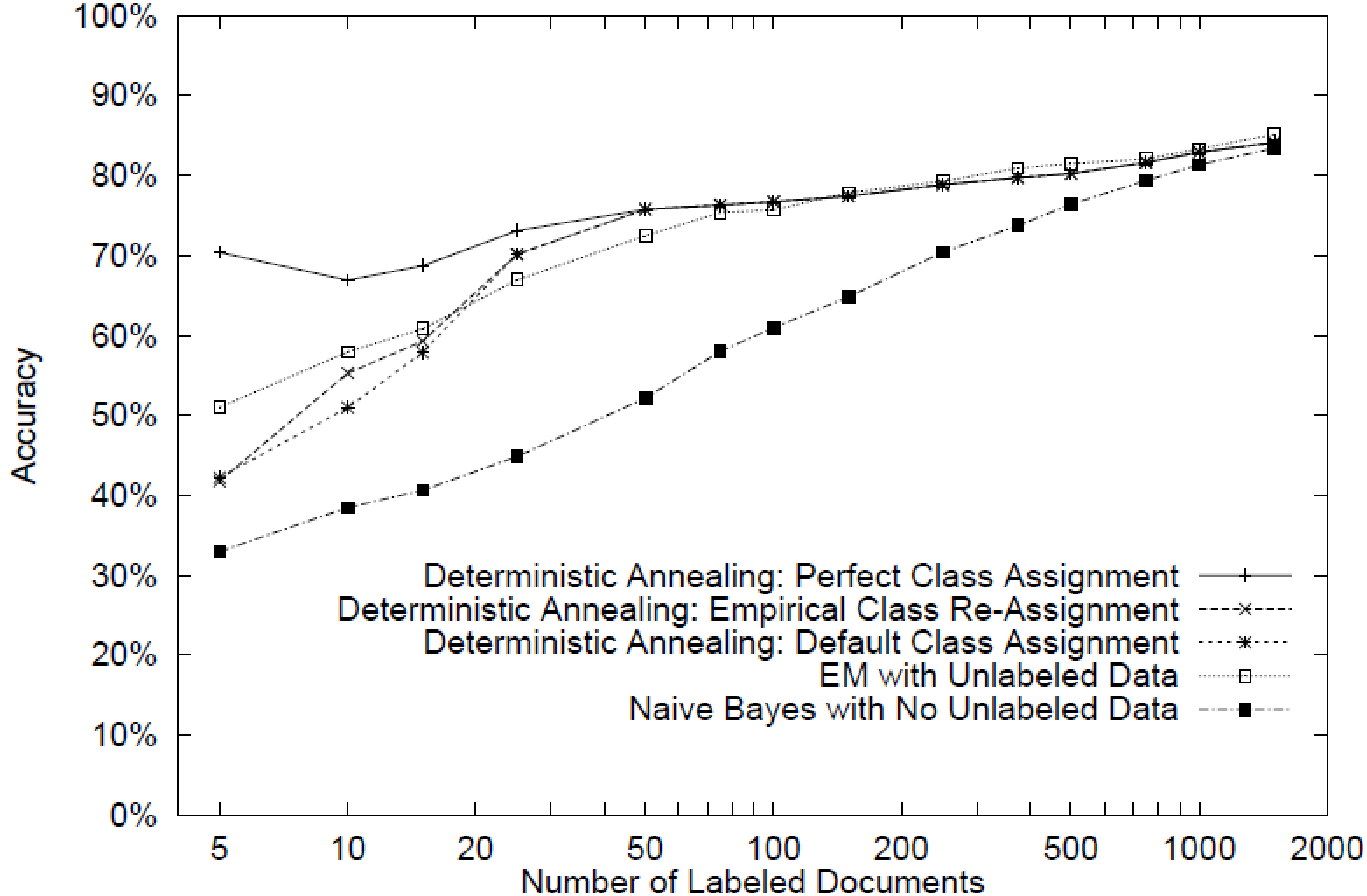
# Problem: local maxima

- "Deterministic Annealing"

$$l(\theta|X,Y) = \sum_{x_i \in X_u} \log \sum_{c_j \in [M]} [\mathrm{P}(c_j|\theta)\mathrm{P}(x_i|c_j;\theta)]^{\beta}$$
$$+ \sum_{x_i \in X_l} \log([\mathrm{P}(y_i = c_j|\theta)\mathrm{P}(x_i|y_i = c_j;\theta)]^{\beta})$$

- Slowly increase $\beta$
- Results: works, but can end up confusing classes (next slide)

# Annealing performance



# Homework #4 (1 of 3)

- What if we don’t know the target classes in advance?
- Example: Google Sets

- Wait until query time to run EM?  Slow.

- Strategy: Learn a NB model in advance, obtain mapping from examples->”classes”

- Then at “query time” compare examples

# Homework #4 (2 of 3)

- Classify noun phrases based on *context* in text
  - E.g. `___ prime minister`   `CEO of ___`
- Model noun phrases (NPs) as P(*z* | *w*):

P(z | Canada) =

| z=1 | 2 | | N |
|---|---|---|---|
| 0.14 | 0.01 | … | 0.06 |

- Experiment with different N
- Query time **input**: "seeds" (e.g., Algeria, UK)
  **Output**: ranked list of other NPs, using KL div.

# Homework #4 (3 of 3)

- Code: written in Java
- You write ~4 lines
  - (important ones)
- Run some experiments

- Homework also has a few written exercises
  - “Big picture”

# Road Map

- Basics of Probability and Statistical Estimation
- Bayesian Networks
- Markov Networks
- Inference
- Learning
  - Parameters, Structure, EM
- **HMMs**