10-708: Probabilistic Graphical Models 10-708, Spring 2018

11 : CRF + Intro to Topic Models

Lecturer: Kayhan Batmanghelich *Scribes: Anqi Yang, Junxian He*

1 Conditional Random Field

1.1 CRF Learning

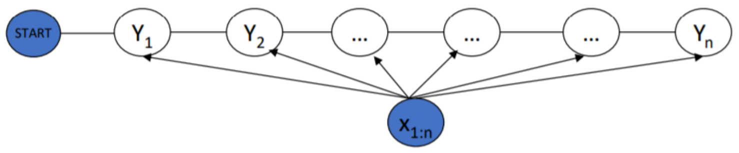

Figure 1: 1-dimensional Conditional Random Field.

As is shown in Equation 1, the conditional probability of chain CRF can be expanded by exponential family, which is composed of a set of binary terms f_k and a set of unary terms g_l.

$$\begin{aligned} P(\mathbf{y}|\mathbf{x}) &= \frac{1}{Z(\mathbf{x},\lambda,\mu)} \exp(\sum_{i=1}^{n}(\sum_{k}\lambda_k f_k(y_i,y_{i-1},\mathbf{x}) + \sum_{l}\mu_l g_l(y_i,\mathbf{x}))) \\ &= \frac{1}{Z(\mathbf{x},\lambda,\mu)} \exp(\sum_{i=1}^{n}(\lambda^T\mathbf{f}(y_i,y_{i-1},\mathbf{x}) + \mu^T\mathbf{g}(y_i,\mathbf{x}))) \\ \text{where} Z(\mathbf{x},\lambda,\mu) &= \sum_{\mathbf{y}}\exp(\sum_{i=1}^{n}(\lambda^T\mathbf{f}(y_i,y_{i-1},\mathbf{x}) + \mu^T\mathbf{g}(y_i,\mathbf{x}))) \end{aligned} \tag{1}$$

Note that $Z(\mathbf{x},\lambda,\mu)$ can be ignored when computing the query $\arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x};\lambda,\mu)$ as there is no $\mathbf{y}$ term in Z, while can not be ignored when computing $\max_{\lambda,\mu}\log P(\mathbf{y}|\mathbf{x};\lambda,\mu)$ as λ and μ are contained in Z.

Given N *i.i.d.* training samples $(\mathbf{x}_d,\mathbf{y}_d)_{d=1}^{N}$, we can use Maximum Likelihood Estimation to learn the parameters λ^*,μ^* such that,

$$
\begin{aligned}
\lambda^*, \mu^* &= \arg\max_{\lambda,\mu} L(\lambda, \mu) = \arg\max_{\lambda,\mu} \prod_{d=1}^{N} P(\mathbf{y}_d|\mathbf{x}_d, \lambda, \mu) \\
&= \arg\max_{\lambda,\mu} \prod_{d=1}^{N} \frac{1}{Z(\mathbf{x}_d, \lambda, \mu)} \exp(\sum_{i=1}^{n} (\lambda^T \mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d) + \mu^T \mathbf{g}(y_{d,i}, \mathbf{x}_d))) \\
&= \arg\max_{\lambda,\mu} \sum_{d=1}^{N} (\sum_{i=1}^{N} (\lambda^T \mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d) + \mu^T \mathbf{g}(y_{d,i}, \mathbf{x}_d)) - \log Z(\mathbf{x}_d, \lambda, \mu)) \qquad (2)
\end{aligned}
$$

Computing the gradient of function 2 w.r.t. λ, we can get

$$
\begin{aligned}
\nabla_\lambda L(\lambda, \mu) &= \sum_{d=1}^{N} (\sum_{i=1}^{n} \mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d) - \sum_{\mathbf{y}} (P(\mathbf{y}|\mathbf{x}_d) \sum_{i=1}^{n} \mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d))) \\
&= \sum_{d=1}^{N} (\sum_{i=1}^{n} \mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d) - \mathbb{E}_{y \sim P(\mathbf{y}|\mathbf{x}_d)} [\mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d)] \qquad (3)
\end{aligned}
$$

where the second term in Equation 3 in an exponential family is the expectation of the sufficient statistics. If we compute this marginal probability by naive summation over y, this computation is intractable. However, we can utilize the structure of this graphical model and compute the expectation of $\mathbf{f}$ over the corresponding marginal probability of neighboring nodes as following,

$$
\begin{aligned}
\sum_{\mathbf{y}} (P(\mathbf{y}|\mathbf{x}_d) \sum_{i=1}^{n} \mathbf{f}(y_i, y_{i-1}, \mathbf{x}_d)) &= \sum_{i=1}^{n} (\sum_{\mathbf{y}} \mathbf{f}(y_i, y_{i-1}, \mathbf{x}_d) P(\mathbf{y}|\mathbf{x}_d) \\
&= \sum_{i=1}^{n} \sum_{y_i, y_{i-1}} \mathbf{f}(y_i, y_{i-1}, \mathbf{x}_d) P(y_i, y_{i-1}|\mathbf{x}_d) \qquad (4)
\end{aligned}
$$

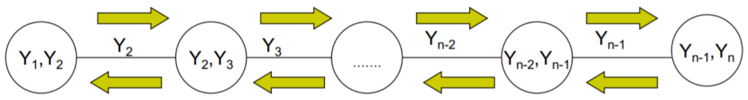

Figure 2: Junction Tree Algorithm for chain CRF.

Basically, we can run a Junction Tree algorithm to compute this marginal probability. As is shown in Figure 2, after one round of forward message passing and one round of backward message passing, each clique y_{i-1}, y_i has the information from both sides and the probability can be written as the calibrated potentials,

$$
P(y_i, y_{i-1}|\mathbf{x}_d) = \frac{\alpha(y_i, y_{i-1})}{\sum_{y_i, y_{i-1}} \alpha(y_i, y_{i-1})} = \alpha'(y_i, y_{i-1}) \qquad (5)
$$

By substituting the feature expectation in Equation 4 using the calibrated potentials in Equation 5, we have

$$
\sum_{y_i, y_{i-1}} \mathbf{f}(y_i, y_{i-1}, \mathbf{x}_d) P(y_i, y_{i-1}|\mathbf{x}_d) = \sum_{y_i, y_{i-1}} \mathbf{f}(y_i, y_{i-1}, \mathbf{x}_d) \alpha'(y_i, y_{i-1}) \qquad (6)
$$

Now we know how to compute $\nabla_\lambda L(\lambda, \mu)$ in Equation 3,

$$\begin{aligned}\nabla_\lambda L(\lambda,\mu) &= \sum_{d=1}^{N}(\sum_{i=1}^{n}\mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d) - \sum_{\mathbf{y}}(P(\mathbf{y}|\mathbf{x}_d)\sum_{i=1}^{n}\mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d))) \\ &= \sum_{d=1}^{N}(\sum_{i=1}^{n}\mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d) - \sum_{y_i, y_{i-1}}\mathbf{f}(y_i, y_{i-1}, \mathbf{x}_d)\alpha'(y_i, y_{i-1}) \end{aligned} \tag{7}$$

And we can learn λ using the gradient descent, similarly for μ,

$$\lambda^{(t+1)} = \lambda^{(t)} + \eta\nabla_\lambda L(\lambda^{(t)}, \mu^{(t)}) \tag{8}$$
$$\mu^{(t+1)} = \mu^{(t)} + \eta\nabla_\mu L(\lambda^{(t)}, \mu^{(t)}) \tag{9}$$

In practice, in order to have a stable estimation for λ and μ without enough observations, we use a Gaussian Regularizer for the parameter vector to improve the generalizability,

$$\lambda^*, \mu^* = \arg\max_{\lambda,\mu}\sum_{d=1}^{N}\log P(\mathbf{y}_d|\mathbf{x}_d, \lambda, \mu) - \frac{1}{2\sigma^2}(\lambda^T\lambda + \mu^T\mu) \tag{10}$$

In practice, we can use the Conjugate Gradient Descent method and Limited Memory Quasi-Newton Methods to accelerate the convergence.

1.2 Case Study in Computer Vision

Apart from CRFs in 1-dimensional, we can also have CRFs for arbitary graph structure. In particular, we are interested in grid CRFs which have important applications in computer vision, such as image restoration, stereo matching, image segmentation and image depth estimation.

1.2.1 Image Segmentation

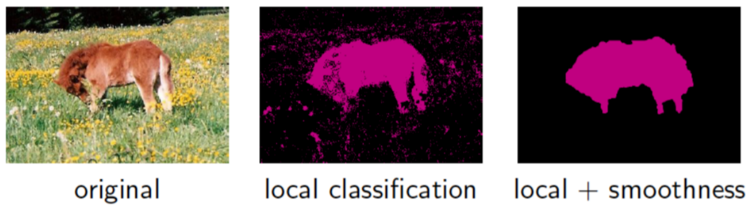

Figure 3: Example of Image Segmentation. The left figure shows the original image, the middle figure shows the classification result for each individual pixel, and the right figure shows the smoothed version of classification.

Viewing each pixel or block as a node, we can construct a 2-dimensional grid CRF to do the image segmentation. Let x denote a pixel, and y_i denote the i-th class, the Gibb distribution of the conditional probability $p(y|x)$ can be written as,

$$P(y|x) = \frac{1}{2}\prod_{i=1}^{n}\Phi(y_i, x)\prod_{i=1}^{n}\Phi(y_i, y_{i-1}) \tag{11}$$

where $\Phi_i(y_i, x) \in \mathbb{R}^{\approx 1000}$ is local image features w.r.t. to the i-th classifier such as bag-of-words features. And $\Phi_{i,j}(y_i, y_j) = \mathbb{1}[y_i = y_j] \in \mathbb{R}^1$ tests whether the neighboring pixels have the same label. Only by maximizing the likelihood of the term $< w_i, \Phi_i(y_i, x) >$, we can learn a local classifier like logistic regression for individual pixel, while jointly maximizing $< w_{ij}, \Phi_{ij}(y_i, y_j) >$ will further penalize the label changes and result in a smoothed version of local cues as is shown in the right of Figure 3. Formally, the Maximum log-Likelihood Estimation of this conditional distribution can be written as,

$$Y^* = \arg\max_{y \in (0,1)^n} [\sum_{i \in S} V_i(y_i, X) + \sum_{i \in S} \sum_{j \in N_i} V_{i,j}(y_i, y_j)] \tag{12}$$

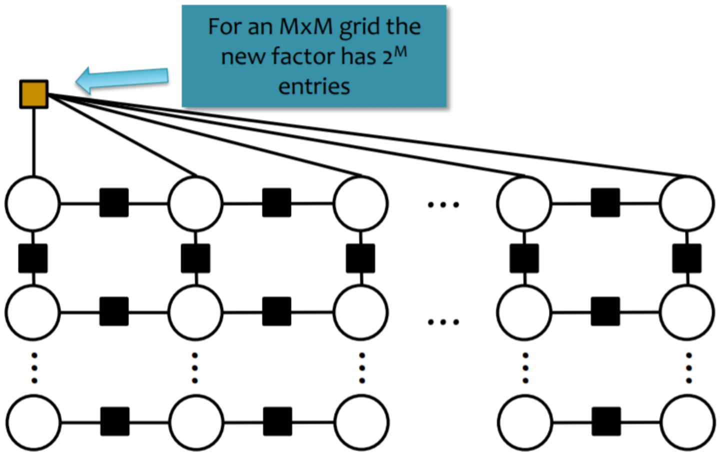

Figure 4: Illustration of Variable Elimination in Grid CRF.

where Y denotes the labels, X is the features, S is the pixels and N_j is the neighbors of pixels i.

Note that for grid CRF, the inference and learning is no longer tractable. Figure 4 illustrates the variable elimination in grid CRF. The original factor graph has factors of size 4. Each time we eliminate a node by marginalizing, one new connection is introduced, which makes the size of factor increases by 2 times. Thus for an $M \times M$ grid, the new factor would have 2^M entries even when eliminating one row of nodes. Apparently, direct variable elimination is too expensive to run, thus we will talk about approximate inference in later lectures.

1.2.2 Pose Estimation

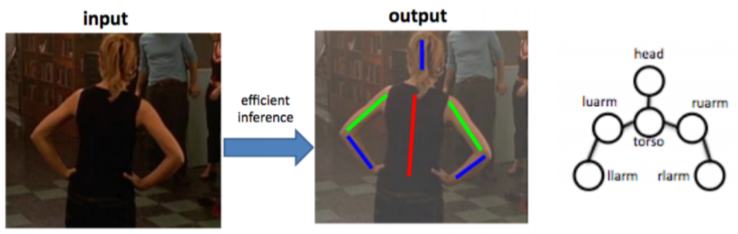

Figure 5: Illustration of Pose Estimation using CRF.

Another application of CRFs in computer vision is pose estimation, where we view each deformable parts of a human body as a random variable, and estimate the location of each deformable parts, as is illustrated in Figure 5.The conditional probability of location l given local feature x of each patch can be written as

$$p(l|x) \propto \exp[\sum_{ij} \theta_{ij}^T \Phi_{ij}(l_i, l_j, x) + \sum_{i} \theta_i^T \Phi_i(l_i, x)] \tag{13}$$

where $\Phi_{ij}(l_i, l_j, x)$ is the prior knowledge of pose, and $\Phi_i(l_i, x)$ is the local representation of features given location classifier l_i. Thus the first term is the penalizer of unrealistic pose and the second term is the local classifier for each part. By Maximum log-Likelihood Estimation of $p(y|x)$, we will be able to learn a cleaned up version of pose estimator.

1.2.3 Object Recognition

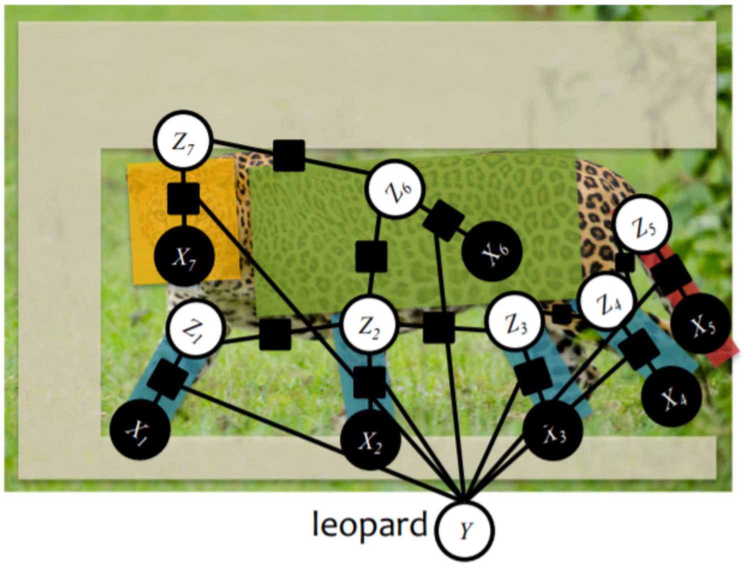

Figure 6: Illustration of Object Recognition using Hidden-state CRFs.

Hidden-state CRFs can be applied to object recognition. The idea is to separate a whole image into parts and inference object class by the information of each parts and the spaticial relationship between these parts. Given an image x and label y, we first preprocess the image into "patches" X_i. For each path X_i, we assign a latent label Z_i to describe the location of object parts (e.g. head, leg, tail, torso, grass). Note that z is not observed in both training and testing. Thus we can introduce factors between X_i and Z_i which describes the location given local features, and factors between hidden variables (Z_i, Z_j), which describes the spaticial relationships exists between different parts, and the factors between (Z_i, Y) which relates the location of each part to the image-level label. We can formulate the joint model as,

$$p_\theta(\mathbf{y}, \mathbf{z}|\mathbf{x}) = \frac{1}{Z(\mathbf{x}, \theta)} \prod_{\alpha} \Phi_\alpha(\mathbf{y}_\alpha, \mathbf{z}_\alpha, \mathbf{x}) \tag{14}$$

As $\mathbf{z}_\alpha$ is unobserved, we need to marginalize it out, the derived marginalized model can be written as,

$$p_\theta(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{z}} p_\theta(\mathbf{y}, \mathbf{z}|\mathbf{x}) \tag{15}$$

We train this model using gradient based method and in each step, the gradient w.r.t. θ is,

$$\begin{aligned}
\frac{dl(\theta|D)}{d\theta} &= \sum_{n=1}^{N} (\mathbb{E}_{z\sim p_\theta(|\mathbf{y}^{(n)})}[f_j(\mathbf{y}^{(n),\mathbf{z}})] - \mathbb{E}_{\mathbf{y},\mathbf{z}\sim p_\theta(;)}[f_j(\mathbf{y}, \mathbf{z})]) \\
&= \sum_{n=1}^{N} \sum_{\alpha} (\sum_{\mathbf{z}_\alpha} p_\theta(\mathbf{z}_\alpha|\mathbf{y}^{(n)}) f_{\alpha,j}(\mathbf{y}_\alpha^{(n)}, \mathbf{z}_\alpha) - \sum_{\mathbf{y}_\alpha, \mathbf{z}_\alpha} p_\theta(\mathbf{y}_\alpha, \mathbf{z}_\alpha) f_{\alpha,j}(\mathbf{y}_\alpha, \mathbf{z}_\alpha))
\end{aligned}$$

where the first term is inference on clamped factor graph and the second term is the inference on full factor graph. This can be optimized in an EM way.

2 Topic Modeling

2.1 Motivation

- Organize a massive unlabeled corpora with thematic categories
- Describe the evolution of those categories over time
- Enable a domain expert to analyze and understand the content
- Find relationships between the categories
- Understand how author ship influences the content

2.2 Overview

Topic modeling is a (usually unsupervised) method of discovery of latent or hidden structure in a corpus. Note that the data for topic modeling is not necessarily text, actually text is a set of words, and topic modeling can also be used to model a set of images.

Applications of topic modeling include Dirichlet-multinomial regression (DMR) topic model [1], map of NIH grants [2], spacial LDA for images [3], etc.

2.3 Review: Latent Dirichlet Allocation (LDA)

2.3.1 Beta-Bernoulli Model

Beta distribution:

$$f(x|\alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1}, \qquad 0 < x < 1 \tag{16}$$

Assume that x is the probability of a coin, then if α is large, then this coin is more fair; if α is smaller, and β is large, then this coin is more biased.

The generative process of Beta-Bernoulli model is as follows:

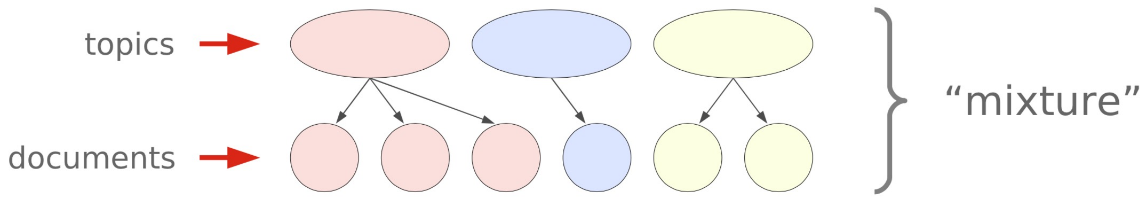

Figure 7: Illustration of mixture model.

1. $\phi \sim \text{Beta}(\alpha, \beta)$ (Draw distribution over words)
2. For each word $n \in \{1, \cdot, N\}$: Draw $x_n \sim \text{Bernoulli}(\phi)$ (Draw word)

An example corpus is a bunch of heads and tails since x_n here is a binary random variable.

2.3.2 Dirichlet-Multinomial Model

An extension of Beta-Bernoulli model is Dirichlet-Multinomial model, Dirichlet distribution is:

$$p(\boldsymbol{\phi}|\boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{k=1}^{K} \phi_k^{\alpha_k - 1}, \quad \text{where} B(\boldsymbol{\alpha}) = \frac{\prod_{k=1}^{K} \Gamma(\alpha_k)}{\Gamma(\sum_{k=1}^{K} \alpha_k)}. \tag{17}$$

Assume all the entries of $\boldsymbol{\alpha}$ are the same, then when α_k is equal to one, the Dirichlet becomes a uniform distribution over $\boldsymbol{\phi}$; if we increase α_k above one, or decrease α_k below one,the Dirichlet over $\boldsymbol{\phi}$ would become sparser.

The generative process of Beta-Bernoulli model is as follows:

1. $\phi \sim \text{Dir}(\boldsymbol{\beta})$ (Draw distribution over words)
2. For each word $n \in \{1, \cdot, N\}$: Draw $x_n \sim \text{Mult}(1, \phi)$ (Draw word)

2.3.3 Dirichlet-Multinomial Mixture Model

The generative process of Dirichlet-Multinomial Mixture model is as follows:

1. For each topic $k \in \{1, \cdots, K\}$:
   - $\boldsymbol{\phi}_k \sim \text{Dir}(\boldsymbol{\beta})$ [draw distribution over words]
2. $\boldsymbol{\theta} \sim \text{Dir}(\boldsymbol{\alpha})$ [draw distribution over topics]
3. For each document $m \in \{1, \cdots, M\}$
   - $z_m \sim \text{Mult}(1, \boldsymbol{\theta})$ [draw topic assignment]
   - For each word $n \sim \{1, \cdots, N_m\}$
     (a) $x_{mn} \sim \text{Mult}(1, \phi_{z_m})$ [draw word]

This generative process defines a mixture of topics, conditioned on which we generate observations, $\boldsymbol{\phi}_k$ represents topic k and is actually a distribution over words, $\boldsymbol{\theta}$ is the topic mixture proportions of document. The generative process is ashwon in shown in Figure 7.

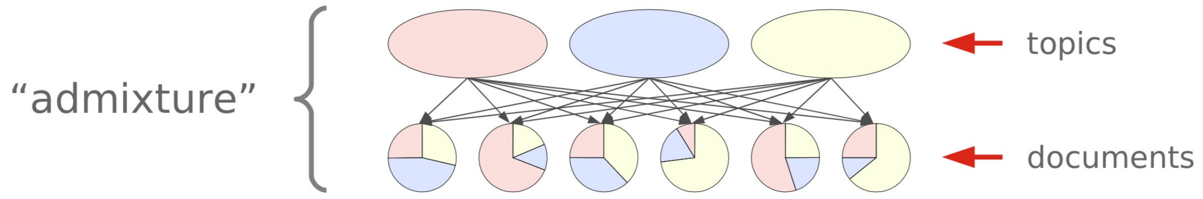

Figure 8: Illustration of admixture model.

2.3.4 Latent Dirichlet Allocation

The generative process of Latent Dirichlet Allocation (LDA) model is:

1. For each topic $k \in \{1, \cdots, K\}$:
   - $\boldsymbol{\phi}_k \sim \text{Dir}(\boldsymbol{\beta})$ [draw distribution over words]
2. For each document $m \in \{1, \cdots, N\}$:
   - $\boldsymbol{\theta}_m \sim \text{Dir}(\boldsymbol{\alpha})$ [draw distribution over topics]
   - For each word $n \in \{1, \cdots, N_m\}$:
     (a) $z_{mn} \sim \text{Mult}(1, \boldsymbol{\theta}_m)$ [draw topic assignment]
     (b) $x_{mn} \sim \boldsymbol{\phi}_{z_{mn}}$ [draw word]

Mixture vs. Admixture (LDA) The difference between them is that for admixture model, each document has its own topic proportions, while the mixture model shares topic proportions, the admixture illustration is shown in Figure 8.

References

[1] David Mimno and Andrew McCallum. Topic models conditioned on arbitrary features with dirichlet-multinomial regression. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, pages 411–418. AUAI Press, 2008.

[2] Edmund M Talley, David Newman, David Mimno, Bruce W Herr II, Hanna M Wallach, Gully APC Burns, AG Miriam Leenders, and Andrew McCallum. Database of nih grants using machine-learned categories and graphical clustering. *Nature Methods*, 8(6):443, 2011.

[3] Xiaogang Wang and Eric Grimson. Spatial latent dirichlet allocation. In *Advances in neural information processing systems*, pages 1577–1584, 2008.