Learning from Multiple Annotators

Gaurav Trivedi

Intelligent Systems Program

gat23@pitt.edu

November 18, 2014

Overview

Learning from Multiple Annotators

Traditional supervised learning

- Ground truth labels are given by a single annotator - *oracle*
- Training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$
  where, $\mathbf{x}_i \in \mathcal{X}$ is a d-dimensional feature vector
  and $y_i \in \mathcal{Y}$ is the known label for it
- The task is to learn a function $f : \mathcal{X} \to \mathcal{Y}$
  which can be used on unseen data

Learning from Multiple Annotators

Traditional supervised learning

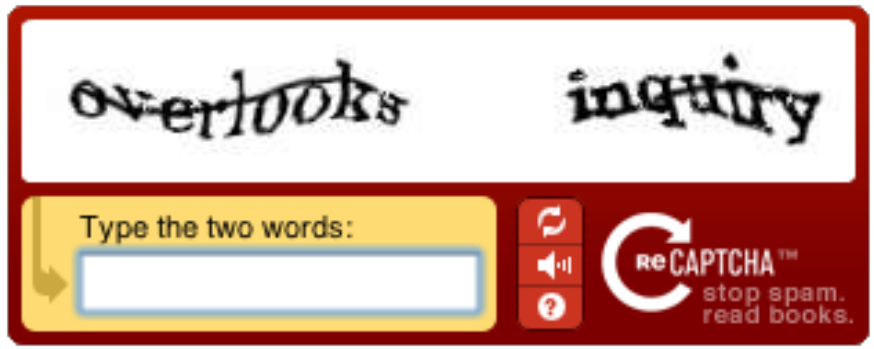
- Ground truth labels are given by a single annotator - *oracle*
- Training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$
  where, $\mathbf{x}_i \in \mathcal{X}$ is a d-dimensional feature vector
  and $y_i \in \mathcal{Y}$ is the known label for it
- The task is to learn a function $f : \mathcal{X} \to \mathcal{Y}$
  which can be used on unseen data

Multiple-annotator learning

- Each example may be labeled by one or more annotators
- Labels may be unreliable (noise)

Everyone has been an annotator!

reCAPTCHA - `www.captcha.net`



Application Scenarios

Quang's review [Quang, 2013] presents the following three scenarios:

- **Each example is labeled by large number of annotators**
  - Labels from a single annotator are unreliable
  - Can we come up with a consensus "true" label?
  - e.g. Crowd-sourcing services like MTurk

Application Scenarios

Quang's review [Quang, 2013] presents the following three scenarios:

- **Each example is labeled by large number of annotators**
  - Labels from a single annotator are unreliable
  - Can we come up with a consensus "true" label?
  - e.g. Crowd-sourcing services like MTurk
- **Different annotators label non-overlapping set of examples**
  - Labeling tasks are expensive and require domain expertise
  - Can we distribute the labeling tasks?
  - e.g. Medical domain training data

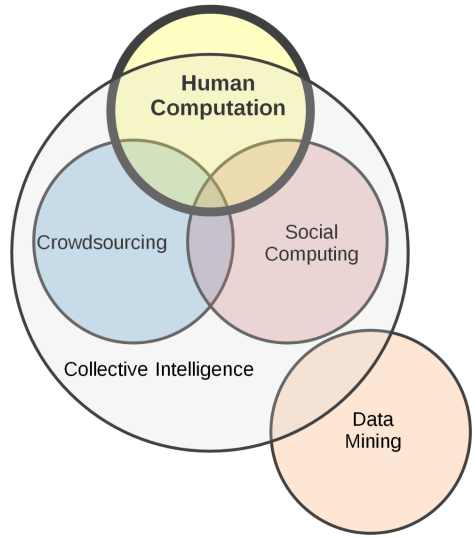
Application Scenarios

Quang's review [Quang, 2013] presents the following three scenarios:

- **Each example is labeled by large number of annotators**
  - Labels from a single annotator are unreliable
  - Can we come up with a consensus "true" label?
  - e.g. Crowd-sourcing services like MTurk
- **Different annotators label non-overlapping set of examples**
  - Labeling tasks are expensive and require domain expertise
  - Can we distribute the labeling tasks?
  - e.g. Medical domain training data
- **Different annotators label overlapping sets of examples**
  - Some examples labeled by one others by many people
  - Can we come up with a consensus model and also explore the relations between different annotators?
  - e.g. Some patients examined by one or several patients

While we are talking about applications...

According to Quinn and Bederson's survey on *Human computation* [Quinn, 2011]:



These applications may fall at the intersection of:

- **Crowdsourcing**
  - outsourcing work to a group in an open call
- **Human computation**
  - extract work that is "difficult for computers"
  - directed by a computational process

Back to the learning process...

For each example i in the training set $\mathcal{D}$,
We don't have the actual label z_i

But, have multiple (possibly noisy) labels
$y_i^1, ..., y_i^M$ provided by M annotators

Learning the "true" label

1. Find "true" labels representative of the provided labels
2. These labels can be then used to learn a predictive model

Learning a consensus model

1. Consensus model is representative of different annotators
2. Can be then applied directly for future predictions

Overview

Learning the "true" labels

- Motivated by the crowd-sourcing applications
- The objective is to find the (true) consensus label, z_i for each example
- We assume the examples are labeled without explicit feature vectors - like we have in many crowdsourcing applications

- The simplest approach would to use a **majority vote**:

  For each example $i \in \{1, 2..., N\}$,

$$z_i = \begin{cases} 1 & \frac{1}{M}\sum_{j=1}^{M} y_i^j > 0.5 \\ 0 & otherwise \end{cases}$$

Problems with Majority Vote

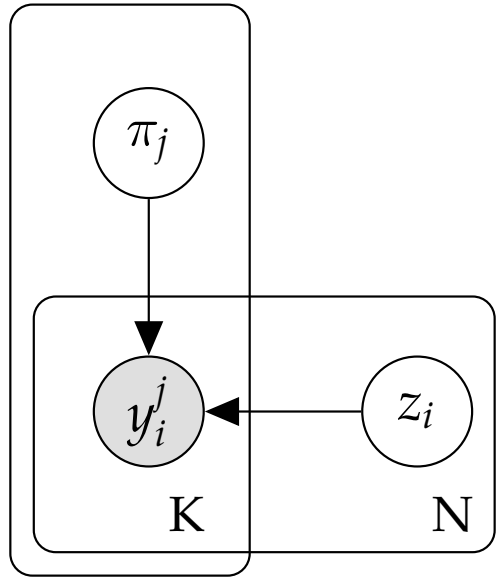
- Assumes that all experts are equally good
- If one reviewer is very reliable and other ones are not, the majority vote would sway the consensus values away from the reliable labels

Problems with Majority Vote

- Assumes that all experts are equally good
- If one reviewer is very reliable and other ones are not, the majority vote would sway the consensus values away from the reliable labels

- What if we introduce weights representing the quality of the reviews?
- This brings us to **Dawid and Skene**'s model [Dawid, 1979].

Dawid and Skene's model



- Again, z_i denotes the hidden true label for example i
- y_i^j denote the label provided by an annotator j
- π_j (hidden) represent the quality of reviews provided by each annotator
  - There can be variables each for modeling accuracy using a confusion matrix
- Use an EM algorithm to learn y_is (E step) and π_ks (M step)

Online Crowdsourcing model

- Imagine a Mechanical Turk like setting where you have access to a large pool of annotators
- The quality of labels varies - good and bad annotators

- Start by seeking a large number of labels from different annotators
- Can we identify annotators providing high quality labels?
- Then we can obtain "true" labels with fewer reliable annotators

Online Crowdsourcing model

- Imagine a Mechanical Turk like setting where you have access to a large pool of annotators
- The quality of labels varies - good and bad annotators

- Start by seeking a large number of labels from different annotators
- Can we identify annotators providing high quality labels?
- Then we can obtain "true" labels with fewer reliable annotators

- Again, we don't really have access to "true" labels!
  - *Welinder and Perona's model*

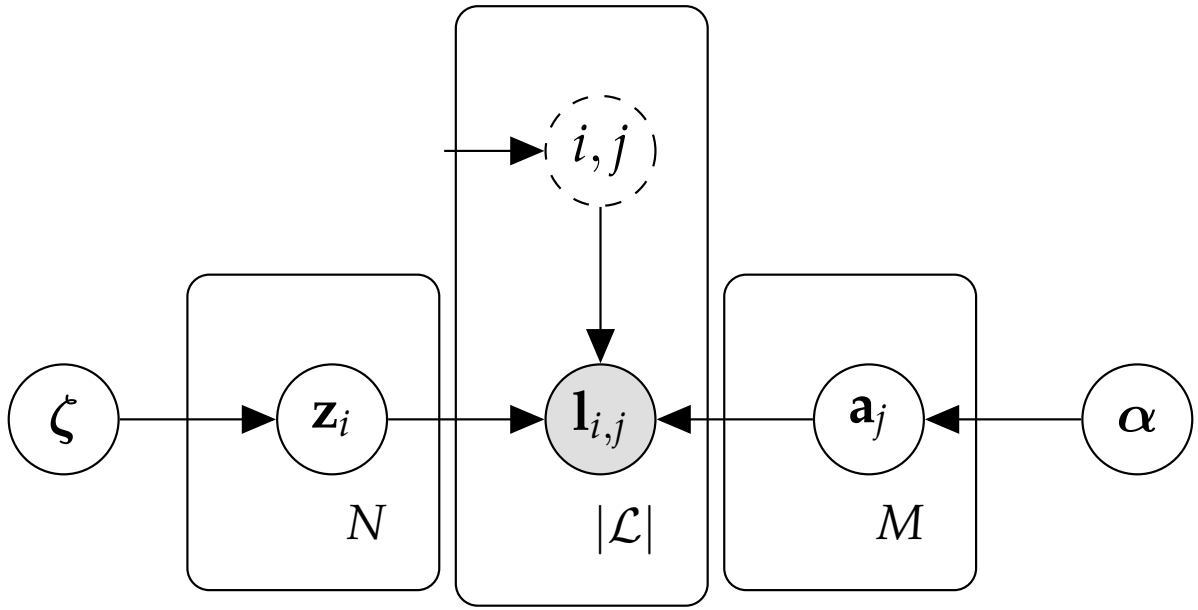
Welinder and Perona's model

- Each example i has an unknown "true" label, $\{\mathbf{z}_i\}_{i=1}^{N}$
  - We can also encode our prior belief using another parameter $\boldsymbol{\zeta}$

- The expertise of M annotators is described by a vector of parameters, $\{\mathbf{a}_j\}_{j=1}^{M}$
  - e.g. $\mathbf{a}_j = a_j$, models the simple accuracy of annotator j
  - Again, we can put another parameter $\boldsymbol{\alpha}$ for the priors

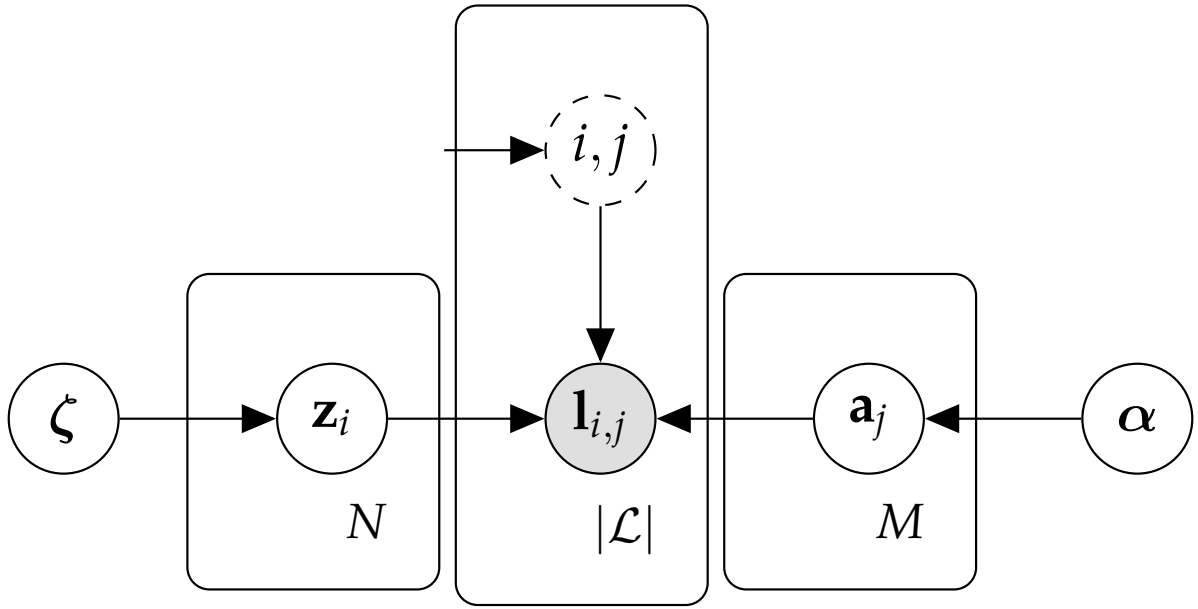
- Each annotator can provide labels for all or a subset of examples.
  - Let each example i be labeled by a set of $\mathcal{A}_i$ annotators
  - It's set of labels are denoted by $\mathcal{L}_i = \{\mathbf{l}_{ij}\}_{j \in \mathcal{A}_i}$

Welinder and Perona's model



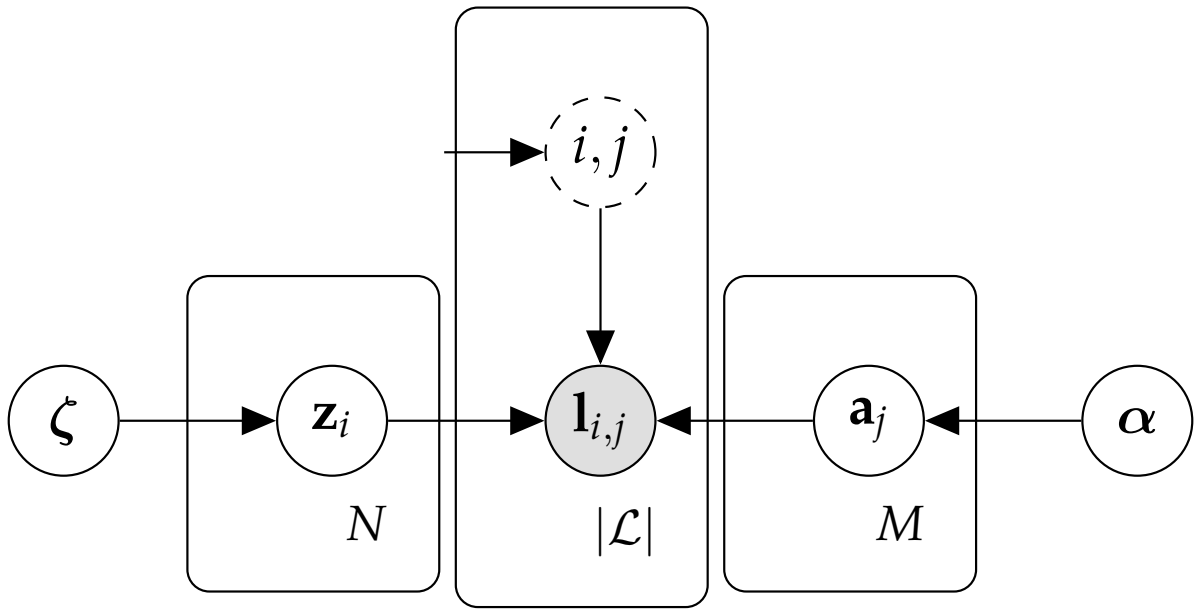
$$p(\mathcal{L}, \mathbf{z}, \mathbf{a}) = \prod_{i=1}^{N} p(\mathbf{z}_i|\zeta) \prod_{j=1}^{M} p(\mathbf{a}_j|\alpha) \prod_{\mathbf{l}_{ij} \in \mathcal{L}} p(\mathbf{l}_{ij}|\mathbf{z}_i, \mathbf{a}_j)$$

Estimating the parameters using EM



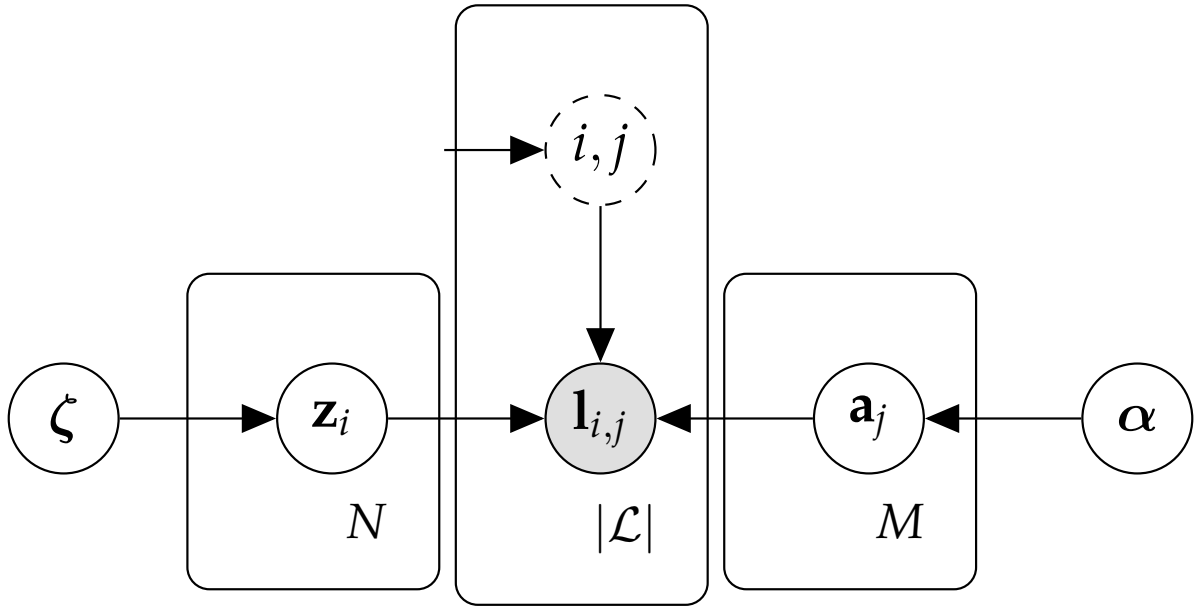
- We observe only $\mathcal{L}$, we need to estimate the hidden variables

Estimating the parameters using EM



- **E-Step** Assume a current estimate for the $\mathbf{a}_j$s, $\hat{\mathbf{a}}$ and compute the posterior for the true labels
  - Use priors ζ for the first iteration

Estimating the parameters using EM

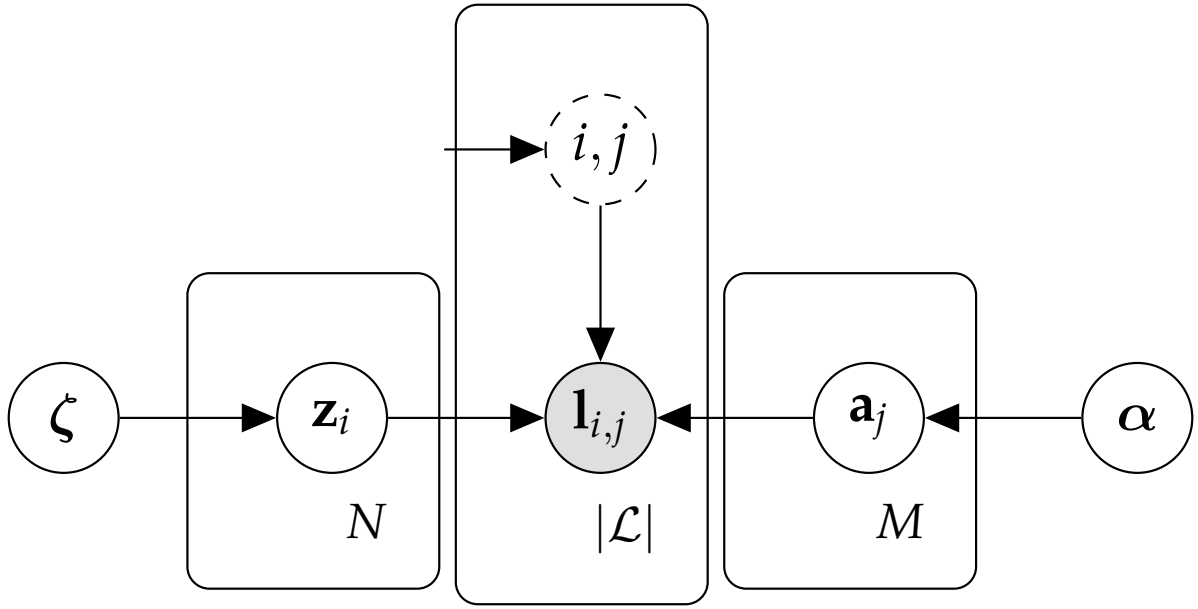


- **E-Step** Assume a current estimate for the $\mathbf{a}_j$s, $\hat{\mathbf{a}}$ and compute the posterior for the true labels

$$\hat{p}(\mathbf{z}) = p(\mathbf{z}|\mathcal{L}, \hat{\mathbf{a}}) \propto p(\mathbf{z})p(\mathcal{L}|\mathbf{z}, \hat{\mathbf{a}}) = \prod_{i=1}^{N} \hat{p}(\mathbf{z}_i)$$

$$\hat{p}(\mathbf{z_i}) = p(\mathbf{z_i}|\boldsymbol{\zeta}) \prod_{j \in \mathcal{A}_i} p(l_{ij}|\mathbf{z_i}_i, \hat{\mathbf{a}}_j)$$

Estimating the parameters using EM



- **M-Step** We need to maximize the the expectation of the log of the posterior on $\mathbf{a}$ using the estimated $\hat{p}(\mathbf{z})$ and $\hat{\mathbf{a}}$ from the previous iteration:

$$a^* = \underset{\mathbf{a}}{\operatorname{argmax}}\, Q(\mathbf{a}, \hat{\mathbf{a}})$$

$$p(\mathbf{a}|\mathbf{z}, \mathcal{L}, \alpha) \propto p(\mathcal{L}|\mathbf{z}, \mathbf{a}) p(\mathbf{a}|\boldsymbol{\alpha})$$

$$Q(\mathbf{a}, \hat{\mathbf{a}}) = \mathbb{E}_{\mathbf{z}}[log\ p(\mathcal{L}|\mathbf{z}, \mathbf{a}) + log\ p(\mathbf{a}|\boldsymbol{\alpha})]$$

Estimating the parameters using EM

- **M-Step**

$$Q(\mathbf{a}, \hat{\mathbf{a}}) = \mathbb{E}_{\mathbf{z}}[log\ p(\mathcal{L}|\mathbf{z}, \mathbf{a}) + log\ p(\mathbf{a}|\boldsymbol{\alpha})]$$

  - Optimization can be carried out for each annotator separately, using only the labels provided by them:

$$Q(\mathbf{a}, \hat{\mathbf{a}}) = \sum_{j=1}^{M} Q_j(\mathbf{a}_j, \hat{\mathbf{a}}_j)$$

  and,

$$Q_j(\mathbf{a}_j, \hat{\mathbf{a}}_j) = log\ p(\mathbf{a_j}|\boldsymbol{\alpha}) + \sum_{i \in \{1,\ldots N\}} \mathbb{E}_{\mathbf{z}_i}[log\ p(\mathbf{l}_{ij}|\mathbf{z}_i, \mathbf{a}_j)]$$

Online Estimation

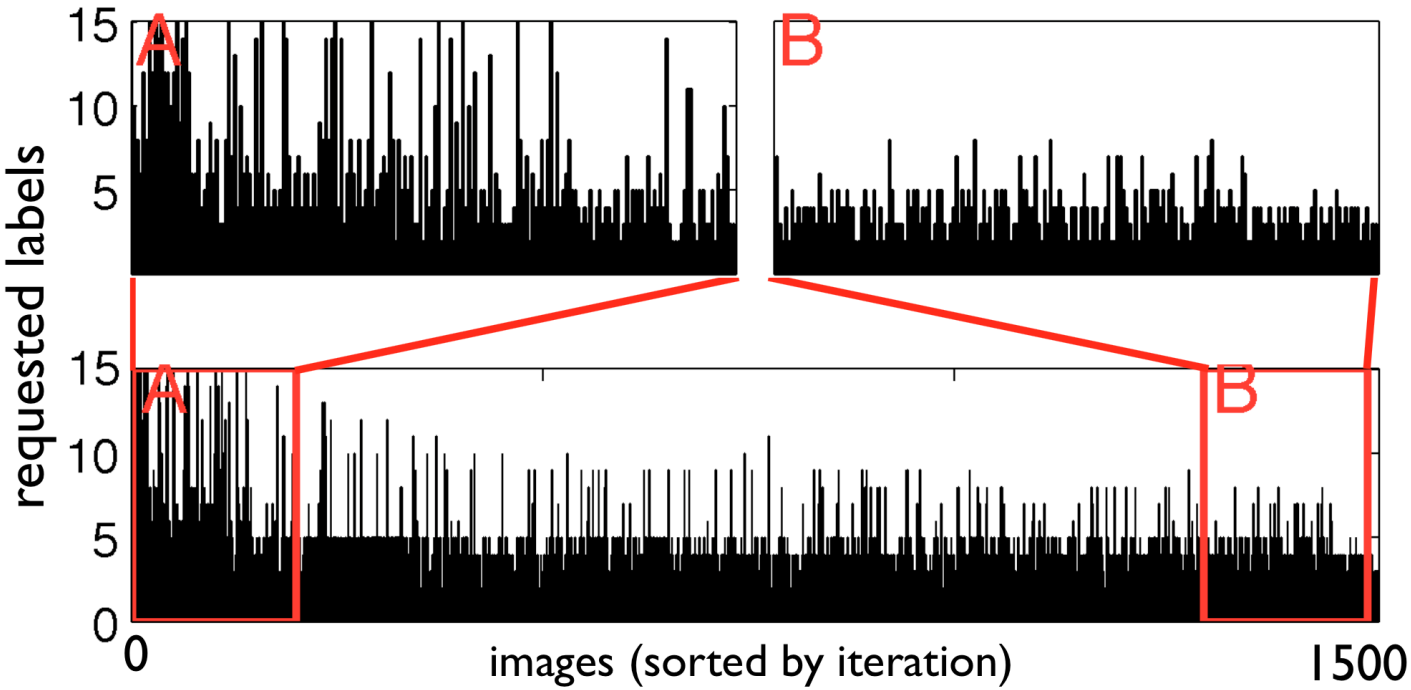
- By looking at $\hat{p}(\mathbf{z})$s, we can estimate how confident we are about a particular label. Also, $\mathbf{a}_j$s can tell us about the performance of the annotators.

- **Label Collection**
  - We can ask for more labels for examples where the target $\mathbf{z}_i$ values are still uncertain

- **Annotator Evaluation**
  - Expert annotators have the variance of their $\mathbf{a_j}$ less than a specific threshold
  - We can give more work to expert annotators and save money as fewer total labels would be required

Remarks

- An example set of MTurk experiments:



- We can make slight modifications to the model to allow different types of annotations: Binary, Multi-valued, and also Continuous labels.

But then again, we are dealing with "human" annotators

[ONLY THE REAL WORD IS NEEDED]

You will be given 2 words: 1 real, 1 fake.

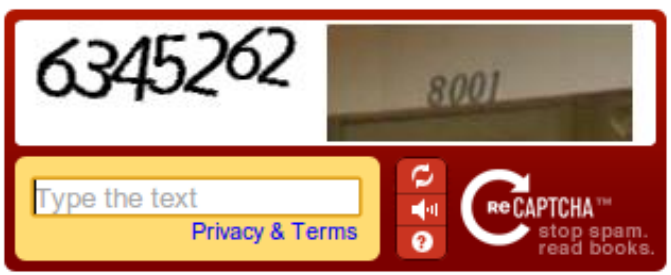
For [REAL FAKE] or [FAKE REAL], you can just type in REAL
accepted.

If it's [LOOKSREAL LOOKSREAL] or [LOOKSFAKE LOOKSF
quicker to just type in both words.
Don't waste precious time deciding which one of them is real.

Use both the appearance and the type of word to identify a fak
Don't rely on just one of them.

[A. IDENTIFYING FAKES WORDS BY TYPE]

[100% FAKE]:
Number
Symbol

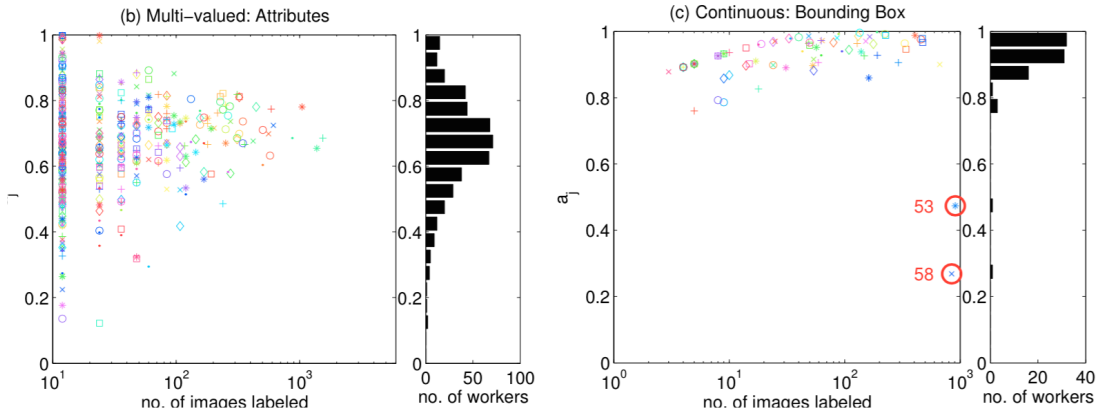


REFRESH DATA

Rank	Name	Avg. Rating	Total Votes
1	moot	90	16,794,368
2	Anwar Ibrahim	47	2,316,378
3	Rick Warren	45	1,902,383
4	Baitullah Mehsud	45	1,902,162
5	Larry Brilliant	44	2,005,310
6	Eric Holder	43	1,808,663
7	Carlos Slim	41	1,852,506
8	Angela Merkel	41	1,634,488
9	Kobe Bryant	39	1,976,880
10	Evo Morales	39	1,477,789
11	Alexander Lebedev	38	824,073
12	Lil' Wayne	37	939,993
13	Sheikh Ahmed bin Zayed Al Nahyan	36	838,578
14	Odell Barnes	35	916,836
15	Tina Fey	33	897,045
16	Hu Jintao	32	928,400
17	Eric Cantor	32	833,208
18	Gamal Mubarak	31	830,677
19	Ali al-Naimi	30	878,743
20	Muqtada al-Sadr	29	810,573
21	Elizabeth Warren	28	2,320,902
22	Manny Pacquiao	27	20,391,818
23	Rain	18	12,762,228
24	Paul Kagame	18	3,890,609
25	Jon Stewart	18	3,152,501

Figure: moot wins, Time Inc. loses
[Music Machinery, 2009]

But then again, we are dealing with "human" annotators



- Annotators want to "optimize" for time and money
- Need to design tasks carefully! [Kittur, 2008]

Overview

Learning the consensus models

- Primary goal is to learn a consensus model that can be used in future for prediction
- Discovering the abilities of the experts comes as a bonus
- We do care about the feature vectors $\mathbf{x}_i$ in this case

- We will cover two models under this:
  - [Raykar, 2010]'s model to learn annotator reliability and the consensus model
  - Learning different expert classification models and finding consensus [Valizadegan, 2013]

Learning from Crowds

- We want to *jointly* learn the consensus model, annotator accuracy and the "true" label
- We measure the performance of an annotators in terms of sensitivity (α) and specificity (β)
- Assume logistic regression for classification (*Can be changed*)
- Annotators are not expected to label all instances. We use EM to estimate them as well

Two Coin Model

- Training set $\mathcal{D} = \{(\mathbf{x}_i, y_i^1, ..., y_i^M)\}_{i=1}^N$
- For each annotator j, let z_i be the actual label for an example

  Sensitivity $\alpha^j = p(y^j = 1 | z_i = 1)$
  Specificity $\beta^j = p(y^j = 0 | z_i = 0)$

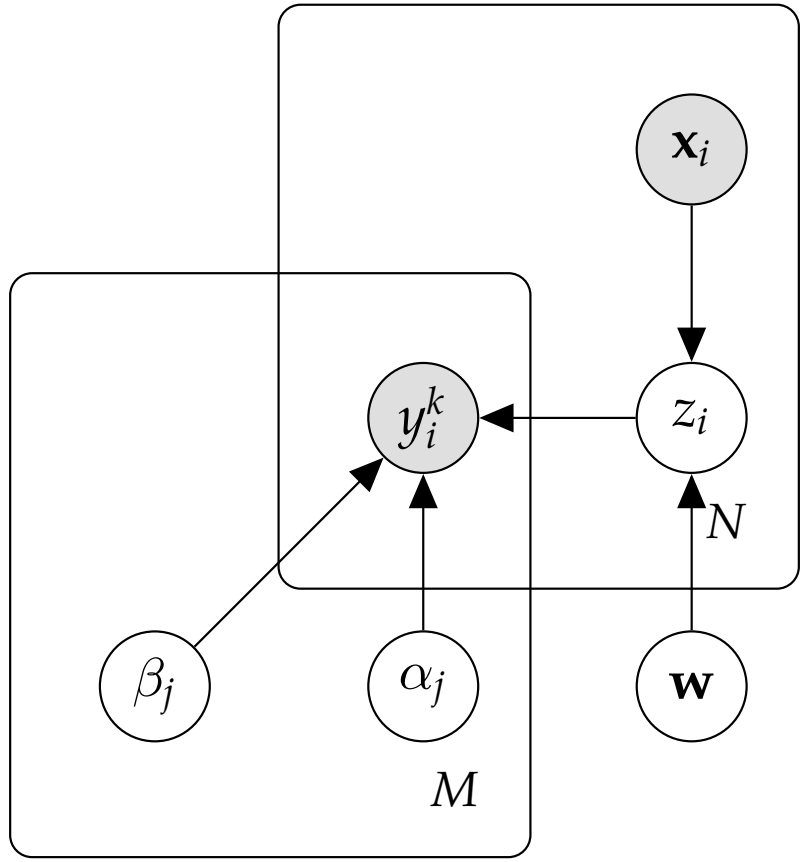
- We assume that α^j and β^j do not depend on the feature vector $\mathbf{x}_i$

Learning Framework

- Training set $\mathcal{D} = \{(\mathbf{x}_i, y_i^1, ..., y_i^M)\}_{i=1}^N$
- The objective is to learn the weight vector $\mathbf{w}$ and the sensitivity $\boldsymbol{\alpha} = [\alpha^1, ...\alpha^M]$ and specificity $\boldsymbol{\beta} = [\beta^1, ...\beta^M]$ of M annotators.
- We will also estimate the "true" labels $z_1, ...z_N$
- Classification is done by a logistic function $P[z_i = 1|\mathbf{x_i}, \mathbf{w}] = \sigma(\mathbf{w}^T\mathbf{x})$

  where, $\sigma(z) = \frac{1}{1+e^{-z}}$

Maximum Likelihood Estimator

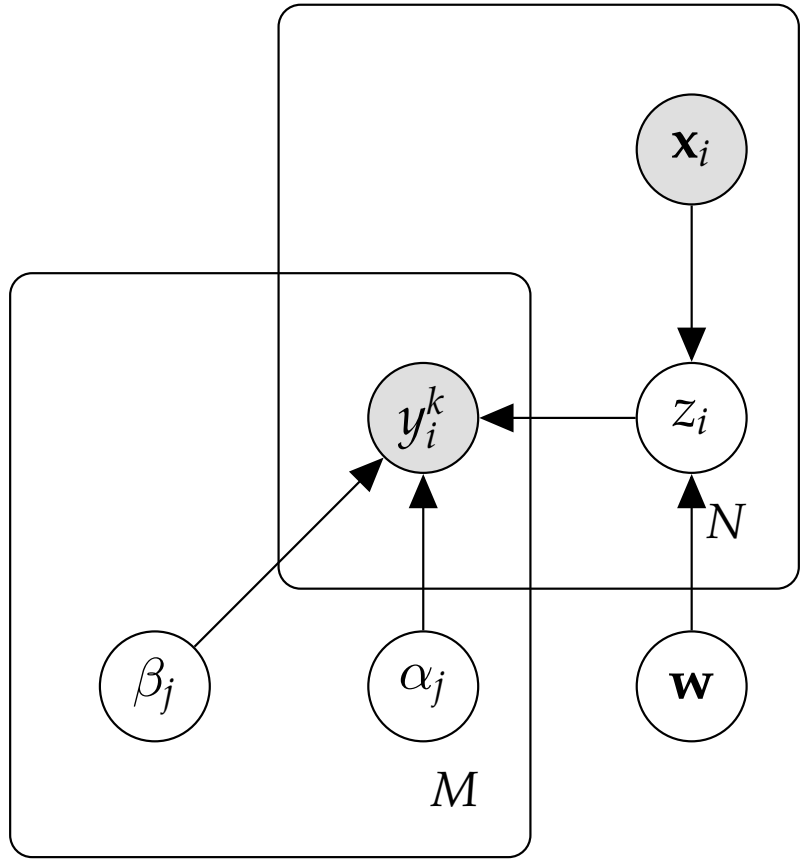


- The likelihood function can be factored as:

$$p[D|\Theta] \propto \prod_{i=1}^{N} p[y_i^1, ...y_i^M|\mathbf{x}_i, \Theta]$$

where, $\Theta = \{\mathbf{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}\}$

Maximum Likelihood Estimator



- The likelihood function can be factored as:

$$p[\mathcal{D}|\Theta] \propto \prod_{i=1}^{N} p[y_i^1, ...y_i^M|z_i = 1, \boldsymbol{\alpha}]p[z_i = 1|\mathbf{x}_i, \mathbf{w}]$$
$$+ p[y_i^1, ...y_i^M|z_i = 0, \boldsymbol{\beta}]p[z_i = 0|\mathbf{x}_i, \mathbf{w}]$$

Maximum Likelihood Estimator

- We also assume that annotators provide labels independently:

$$p[y_i^1, ...y_i^M | z_i = 1, \boldsymbol{\alpha}] = \prod_{j=1}^{M} p[y_i^j | z_i = 1, \alpha^j] = \prod_{j=1}^{M} [\alpha^j]^{y_i^j} [1 - \alpha^j]^{1 - y_i^j}$$

$$p[y_i^1, ...y_i^M | z_i = 0, \boldsymbol{\beta}] = \prod_{j=1}^{M} [\beta^j]^{1 - y_i^j} [1 - \beta^j]^{y_i^j}$$

Maximum Likelihood Estimator

- Therefore, the likelihood can be written as:

$$p[\mathcal{D}|\Theta] \propto \prod_{i=1}^{N} [a_i p_i + b_i(1 - p_i)],$$

where,

$$p_i = \sigma(\mathbf{w}^T \mathbf{x})$$

$$a_i = \prod_{j=1}^{M} [\alpha^j]^{y_i^j} [1 - \alpha^j]^{1 - y_i^j}$$

$$b_i = \prod_{j=1}^{M} [\beta^j]^{1 - y_i^j} [1 - \beta^j]^{y_i^j}$$

- $\hat{\Theta}_{ML} = \{\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}, \hat{\mathbf{w}}\} = \text{argmax}_\Theta \, log \, p[\mathcal{D}|\Theta]$

Estimating the parameters using EM

$$p[\mathcal{D}|\Theta] \propto \prod_{i=1}^{N}[a_i p_i + b_i(1-p_i)]$$

- Now, if we consider the "true" labels $\mathbf{z} = [z_1, ...z_N]$ as hidden data.
- So, the complete likelihood can be written as:

$$p[\mathbf{z}, \mathcal{D}|\Theta] \propto \prod_{i=1}^{M}[a_i p_i]^{z_i} + [b_i(1-p_i)]^{(1-z_i)}$$

$$log\ p[\mathbf{z}, \mathcal{D}|\Theta] \propto \sum_{i=1}^{M} z_i log\ a_i p_i + (1-z_i) log\ b_i(1-p_i),$$

Estimating the parameters using EM

- **E-Step** Assume a current estimate for the $\mathbf{z}_j$s, $\hat{\mathbf{z}}$
  - We can use majority votes as an initialization for $\hat{\mathbf{z}}$

$$\mathbb{E}\{log\ p[\mathbf{z}, \mathcal{D}|\Theta]\} \propto \sum_{i=1}^{M} \hat{z}_i\ log\ a_i p_i + (1 - \hat{z}_i)\ log\ b_i(1 - p_i),$$

$$\begin{aligned} \hat{z}_i &\propto p[y_i^1, ..., y_i^M | z_i = 1, \Theta] p[y_i = 1 | \mathbf{x}_i, \Theta] \\ &= \frac{a_i p_i}{a_i p_i + b_i(1 - p_i)} \end{aligned}$$

Estimating the parameters using EM

- **M-Step** Based on the current estimate for $\hat{\mathbf{z}}$, we now estimate Θ by maximizing $Q(\Theta|\hat{\Theta})$ given the previous estimate.

- We have a closed form solution for α^j and β^j:

$$\alpha^j = \frac{\sum_{i=1}^N \hat{z}_i z_i^j}{\sum_{i=1}^N \hat{z}_i}, \qquad \beta^j = \frac{\sum_{i=1}^N (1-\hat{z}_i)(1-z_i^j)}{\sum_{i=1}^N (1-\hat{z}_i)}$$

- But for $\mathbf{w}$, we must use a gradient-ascent based optimization.

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \mathbf{H}^{-1}\mathbf{g}$$

where, $\mathbf{g}$ is the gradient vector and $\mathbf{H}$ is the Hessian matrix (See [Raykar, 2010])

Remarks and special cases

Not using features

- If we remove the features $\mathbf{x}_i$ from the model, we'll obtain a result similar to [Dawid, 1979], [Welinder, 2010].

Using Bayesian priors

- We may want to trust a particular expert more than the others
- We can impose beta priors for sensitivity and specificity
- Similarly, we may also assume a zero mean Gaussian prior on the weights $\mathbf{w}$ with an inverse covariance matrix $\mathbf{\Gamma}$ for precision
  - This acts as a L2 regularizer
- We can derive the EM estimates while assuming these priors as well

Remarks and special cases

Estimating gold-standard

- Similar to [Welinder, 2010], we can estimate the gold standard by fixing threshold values for $\hat{z}_i$s

Intuitive interpretation of the estimated label

$$logit(\hat{z}_i) = log\ (odds) = log\ \frac{p[z_i = 1|y_i^1, ...y_i^M, \mathbf{x}_i, \Theta]}{p[z_i = 0|y_i^1, ...y_i^M, \mathbf{x}_i, \Theta]}$$

$$= \mathbf{w}^T \mathbf{x}_i + \sum_{j=1}^{M} y_i^j [logit(\alpha^j) + logit(\beta^j)] + constant...$$

- Thus, we have weighted linear combination of the labels

Remarks and special cases

Multi-class classification

- This model can also be extended for multi-class labels
- We will have different sensitivity and specificity parameters for each class
- The "indicator" exponents (z_is) must be replaced by a delta function, $\delta(u, v) = 1$, if $u = v$ and 0 otherwise
- Priors can be modeled by a Dirichlet function

Ordinal Regression

- Convert the ordinal data into a series of binary data
- Use multi-class approach

Regression

- Let $y_i^j \in \mathbb{R}$ be the continuous target value for instance i by the j annotator
- Use a Gaussian noise model with mean as z_i and inverse-variance (precision) τ^j:

$$p[y_i^j | z_i, \tau^j] = \mathcal{N}(y_i^j, 1/\tau^j)$$

- We assume the target value is given by a linear regression model with additive Gaussian noise:

$$z_i = \mathbf{w}^T x_i + \epsilon$$

where, ϵ is a zero-mean Gaussian random variable with precision Υ.

$$p[z_i | \mathbf{x}_i, \mathbf{w}, \Upsilon] = \mathcal{N}(z_i | \mathbf{w}^T \mathbf{x_i}, 1/\Upsilon)$$

Regression

- Combining the annotator and regression models, we get:

$$p[y_i^j|\mathbf{x}_i, \mathbf{w}, \tau^j, \Upsilon] = \mathcal{N}(y_i^j|\mathbf{w}^T\mathbf{x_i}, 1/\tau^j + 1/\Upsilon)$$

where, the new precision term (λ) can be written as $1/\lambda^j = 1/\tau^j + 1/\Upsilon$

$$p[y_i^j|\mathbf{x}_i, \mathbf{w}, \lambda^j] = \mathcal{N}(y_i^j|\mathbf{w}^T\mathbf{x_i}, 1/\lambda^j)$$

Learning Framework

- Training set $\mathcal{D} = \{(\mathbf{x}_i, y_i^1, ..., y_i^M)\}_{i=1}^N$
- The objective is to learn the weight vector $\mathbf{w}$ and the precision $\lambda = [\lambda^1, ...\lambda^M]$ of M annotators.

Maximum Likelihood Estimator

- The likelihood function can be factored as:

$$p[D|\Theta] \propto \prod_{i=1}^{N} p[y_i^1, ...y_i^M|\mathbf{x}_i, \Theta]$$

  where, $\Theta = \{\mathbf{w}, \boldsymbol{\lambda}\}$

- Putting in the Gaussian model:

$$p[D|\Theta] \propto \prod_{i=1}^{N}\prod_{j=1}^{M} \mathcal{N}(y_i^j|\mathbf{w}^T\mathbf{x_i}, 1/\lambda^j)$$

- $\hat{\Theta}_{ML} = \{\hat{\boldsymbol{\lambda}}, \hat{\mathbf{w}}\} = \text{argmax}_\Theta \, log \, p[\mathcal{D}|\Theta]$

Maximum Likelihood Estimator

- By equating the gradient of the log-likelihood to zero, we get:

$$\frac{1}{\hat{\lambda}^j} = \frac{1}{N}\sum_{i=1}^{N}(y_i^j - \hat{\mathbf{w}}^T\mathbf{x}_i)^2$$

$$\hat{\mathbf{w}} = (\sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i^T)^{-1}\sum_{i=1}^{N}\mathbf{x_i}\frac{\sum_{j=1}^{M}\hat{\lambda}^j y_i^j}{\sum_{j=1}^{M}\hat{\lambda}^j}$$

- We iterate these two steps until convergence
- Once we have Θ, we can also estimate the true values of z_is

Limitations

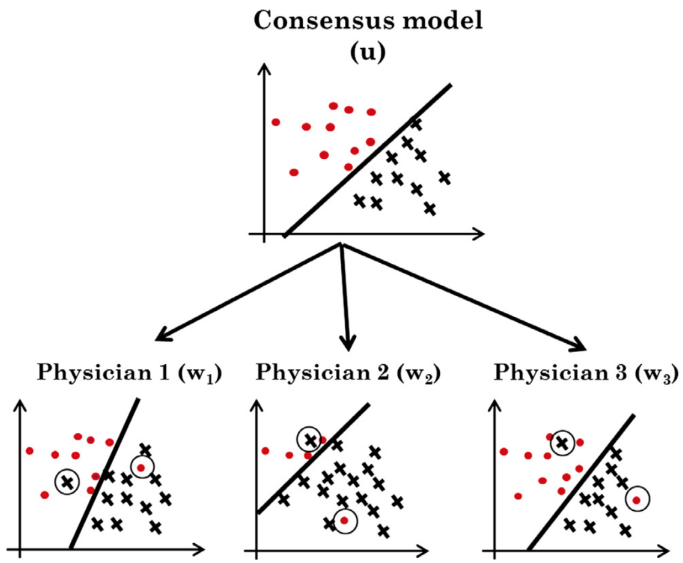
- The model does not estimate the difficulty of the training instance
  - More parameters may be added to capture the difficulty of an instance
- The assumption that sensitivity and specificity are not dependent on the the feature vector $\mathbf{x}_i$ may not very accurate:

$$p[\mathcal{D}|\Theta] \propto \prod_{i=1}^{N} p[y_i^1, ...y_i^M|z_i = 1, \boldsymbol{\alpha}]p[z_i = 1|\mathbf{x}_i, \mathbf{w}] + p[y_i^1, ...y_i^M|z_i = 0, \boldsymbol{\beta}]p[z_i = 0|\mathbf{x}_i, \mathbf{w}]$$

As a result we may need to add another dependency on $\mathbf{x_i}$ thereby increasing the number of parameters to be learned

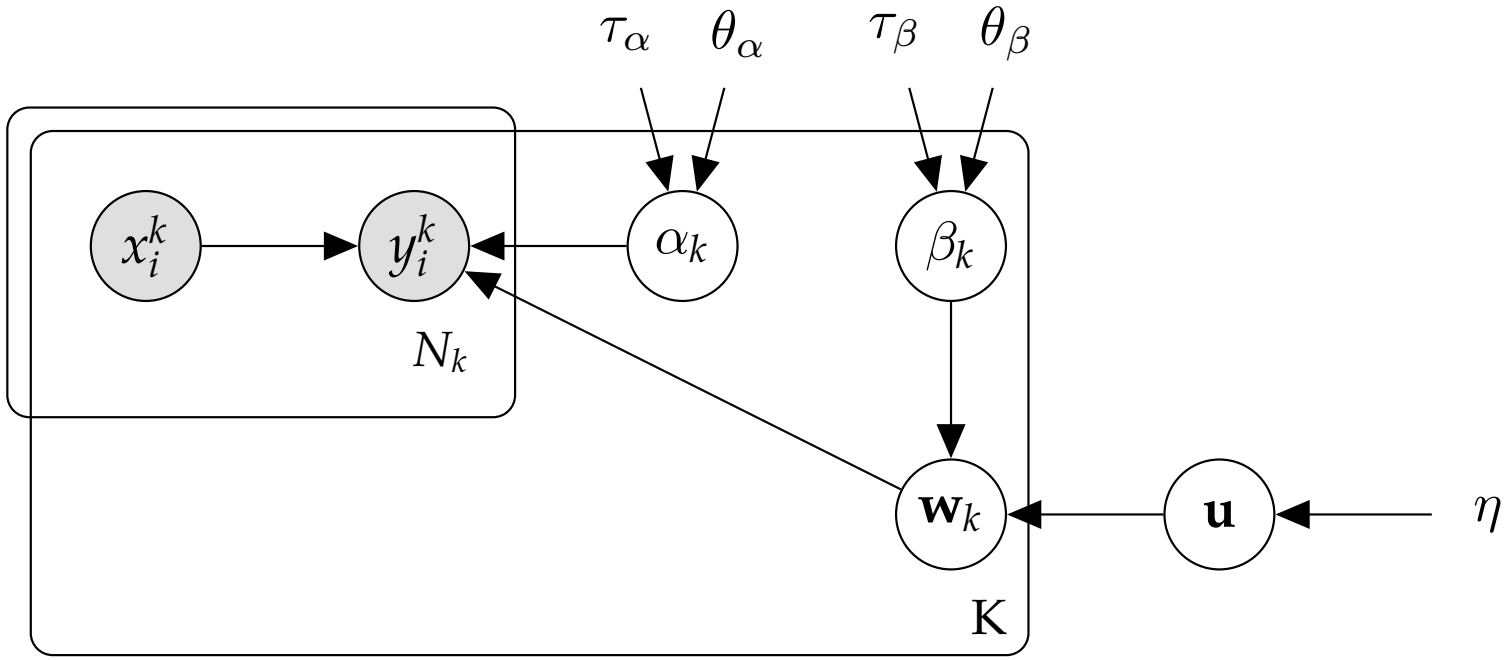
Learning classification models from multiple experts [Valizadegan, 2013]

- This is a multi-expert framework that builds:
  1. a *consensus model* representing the classification model that the experts converge to
  2. *individual expert models* representing the class label decisions exhibited by individual experts



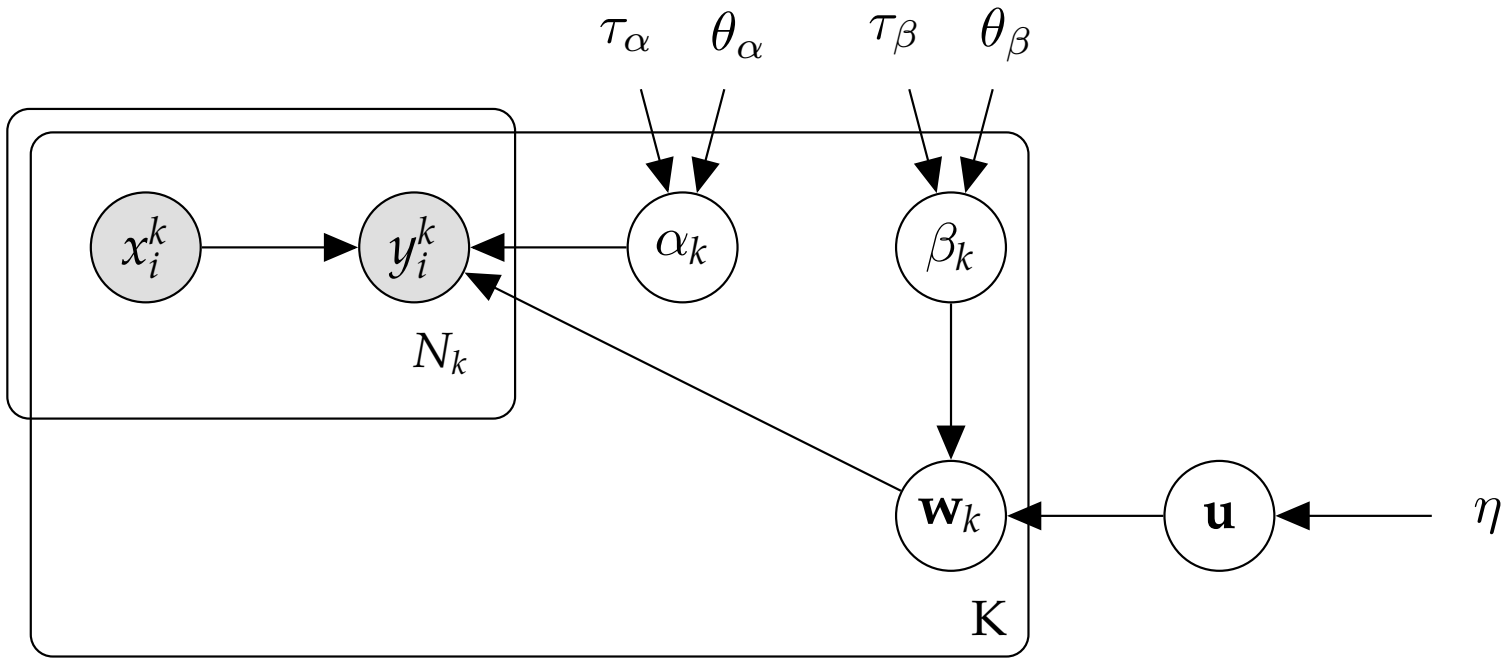
- An important difference from the [Raykar, 2010] model here is that there is more flexibility for the experts pick examples to label

Multi-expert Model



- Consensus models has weights $\mathbf{u}$ and individual expert models have $\mathbf{w}_k$
- α_k is the self-consistency parameter
- β_k is the consensus-consistency parameter. It models the differences in the knowledge of expertise of experts
- Both α_k and β_k may have Gamma priors with two hyper-parameters $\{\tau, \theta\}$

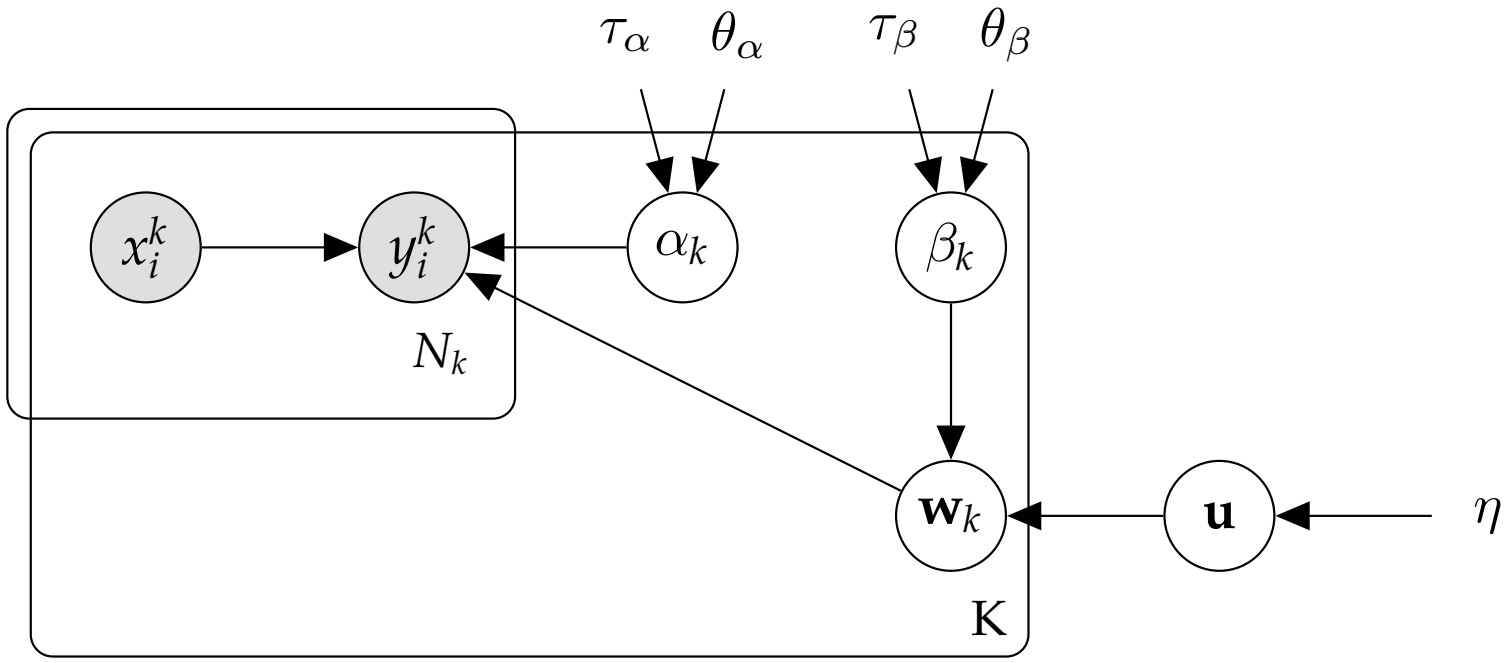
Multi-expert Model



- Then consensus model weights are defined by a Gaussian distribution with zero means

$$p(\mathbf{u}|\mathbf{0}_d, \eta) = \mathcal{N}(\mathbf{0}_d, \eta^{-1}\mathbf{I}_d)$$

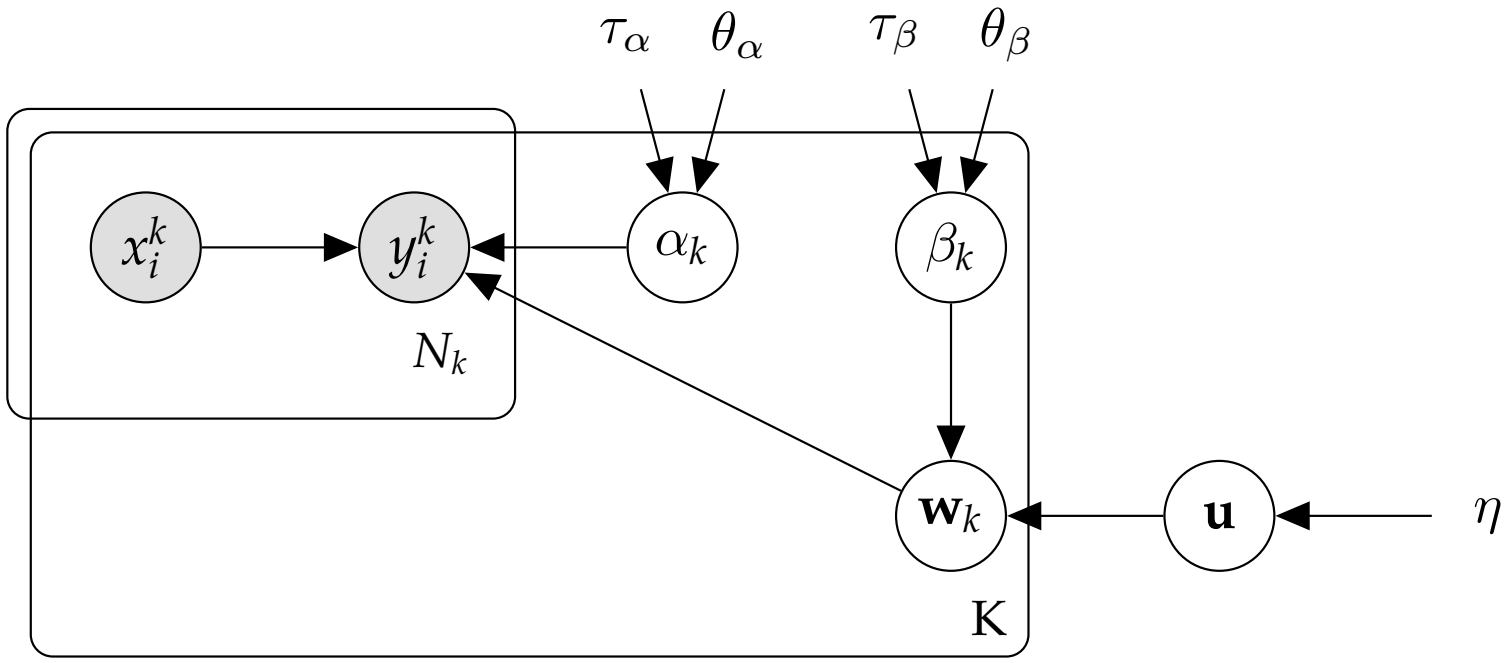
Multi-expert Model



- Expert-specific models are noise corrupted versions of the Gaussian models:

$$p(\mathbf{w}_k|\mathbf{u}, \beta_k) = \mathcal{N}(\mathbf{u}, \beta^{-1}\mathbf{I}_d)$$

Multi-expert Model



- Finally, the parameters $\mathbf{w}_k$ of the expert model relate examples $\mathbf{x}$ to annotator labels:

$$p(y_i^k|\mathbf{x}_i^k, \mathbf{w}_k, \alpha_k) = \mathcal{N}(\mathbf{w}_k^T \mathbf{x}_i^k, 1/\alpha)$$

Optimization

- We take the negative logarithm of the joint given the matrix of examples and their labels provided by experts
  - It now becomes a minimization problem

- Also the squared error term $\|y_i^k - \mathbf{w}_k^T \mathbf{x}_i^k\|^2$ in the objective function is replaced by a hinge loss: $max(0, 1 - y_i^k \mathbf{w}_k^T \mathbf{x}_i^k)$
  - This adds a new set of parameter ϵ_i^k

Optimization

- The objective function is optimized using the alternative optimization approach
  - Split the hidden variables into two: $\{\alpha, \beta\}$ and $\{\mathbf{u}, \mathbf{w}\}$
  - Consider $\{\alpha_k, \beta_k\}$ are considered constants, learn $\{\mathbf{u}, \mathbf{w}_k\}$
  - Then fix $\{\mathbf{u}, \mathbf{w}_k\}$ and compute $\{\alpha_k, \beta_k\}$ by taking derivatives with respect to them. This results in closed form solutions:

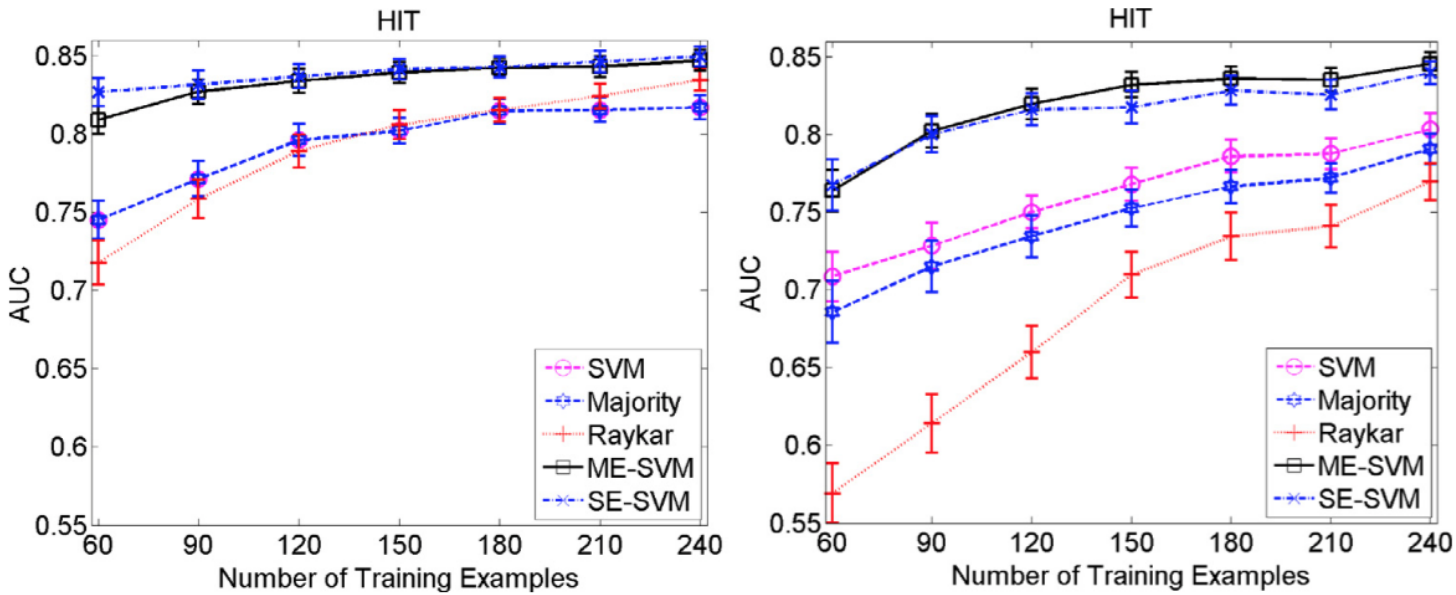
$$\alpha_k = \frac{2(n_k + \theta_{\alpha_k} - 1)}{\sum_{y_i^k=1} \epsilon_i^k + 2\tau_{\alpha_k}}$$

$$\beta_k = \frac{2\theta_{\beta_k}}{\|\mathbf{w}_k - \mathbf{u}\|^2 + 2\tau_{\beta_k}}$$

  - $1/\alpha_k \propto$ the amount of misclassification of examples by expert k with their own model
  - $1/\beta_k \propto$ the difference with the consensus model

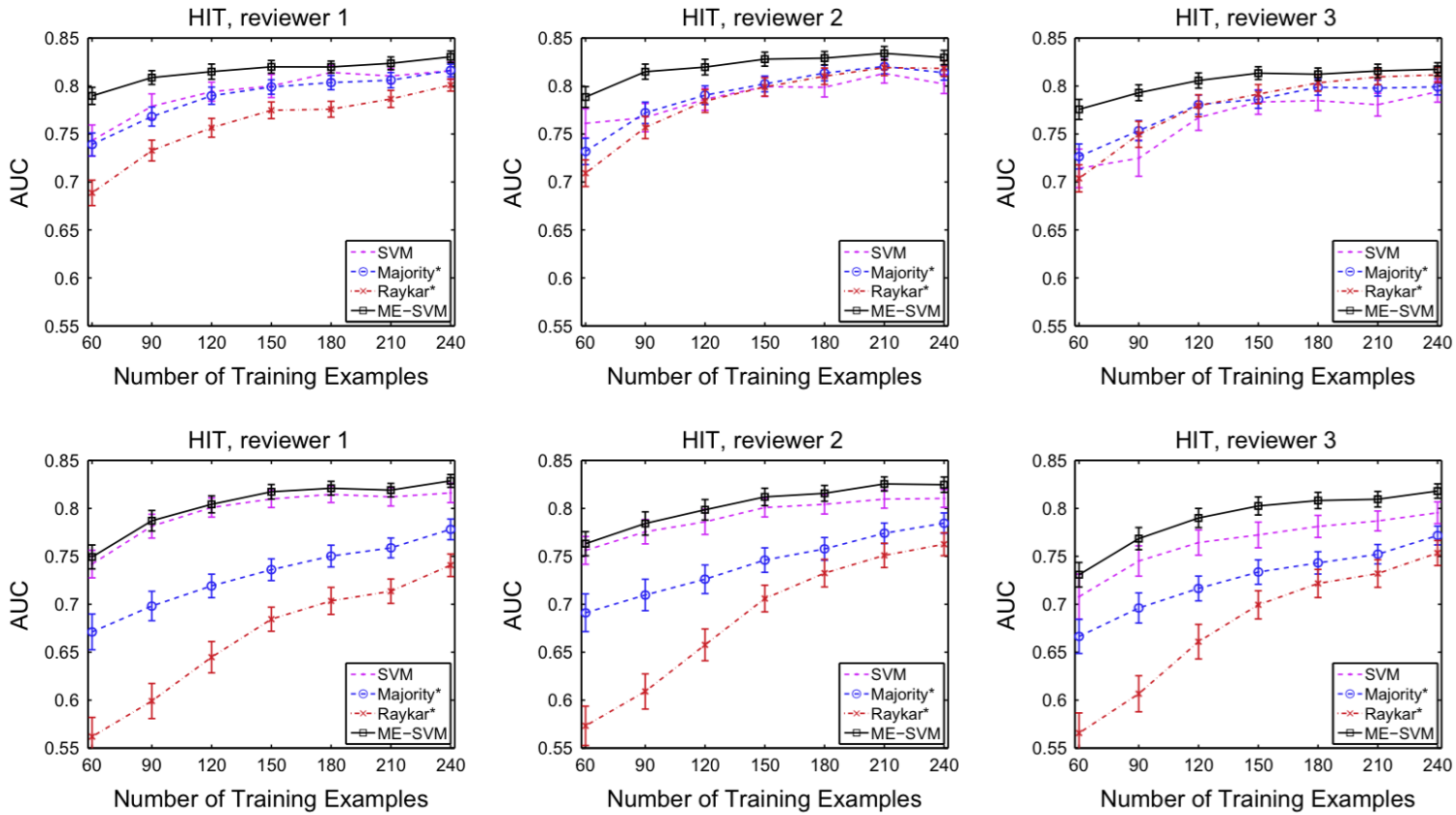
Experimental Results

- Results of the consensus model when every example is labeled by just one expert (left) vs. when all three experts provide labels



Experimental Results

- Expert-specific models



Overview

References

[Dawid, 1979] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied Statistics*, 28(1):20-28, 1979.

[Kittur, 2008] Aniket Kittur, Ed H. Chi, and Bongwon Suh. 2008. Crowdsourcing user studies with Mechanical Turk. In Proceedings of the *SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, New York, NY, USA, 453-456.

[Music Machinery, 2009] moot wins, Time Inc. loses (April 2009). Retrieved from `http://musicmachinery.com/2009/04/27/moot-wins-time-inc-loses/`.

[Quang, 2013] Quang Nguyen (2013). A short review of learning with multiple annotators (Section from Q. Nguyen's thesis proposal).

[Quinn, 2011] Alexander J. Quinn and Benjamin B. Bederson. 2011. Human computation: a survey and taxonomy of a growing field. In Proceedings of the *SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 1403-1412.

References

[Raykar, 2010] Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. 2010. Learning From Crowds. *Journal of Machine Learning Research* 11 (August 2010), 1297-1322.

[Valizadegan, 2013] Valizadegan, Hamed et al. Learning classification models from multiple experts. *Journal of Biomedical Informatics*, Volume 46 , Issue 6 , 1125 - 1135

[Welinder, 2010] Peter Welinder and Pietro Perona. Online crowdsourcing: rating annotators and obtaining cost-effective labels. Workshop on Advancing Computer Vision with Humans in the Loop (ACVHL), *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.