Active Learning in Graph-Based Semi-Supervised Learning

Kevin Miller

University of California, Los Angeles

March 26, 2021

Overview

Observe *labeled data* $\mathcal{D}_\ell = \{(\mathbf{x}_i, y_i)\}_{i \in \mathcal{L}}$ and *unlabeled data* $\mathcal{X}_\mathcal{U} = \{\mathbf{x}_j\}_{j \in \mathcal{U}}$.

- $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} = \mathcal{X}_\mathcal{L} \cup \mathcal{X}_\mathcal{U}$
- $\mathcal{L}$: labeled indices
- $\mathcal{U}$: unlabeled indices
- $Z = \mathcal{L} \cup \mathcal{U}$

Semi-Supervised Learning

From the given data, can we accurately infer the labelings on $\mathcal{U}$?

Setup

Observe *labeled data* $\mathcal{D}_\ell = \{(\mathbf{x}_i, y_i)\}_{i \in \mathcal{L}}$ and *unlabeled data* $\mathcal{X}_\mathcal{U} = \{\mathbf{x}_j\}_{j \in \mathcal{U}}$.

- $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} = \mathcal{X}_\mathcal{L} \cup \mathcal{X}_\mathcal{U}$
- $\mathcal{L}$: labeled indices
- $\mathcal{U}$: unlabeled indices
- $Z = \mathcal{L} \cup \mathcal{U}$

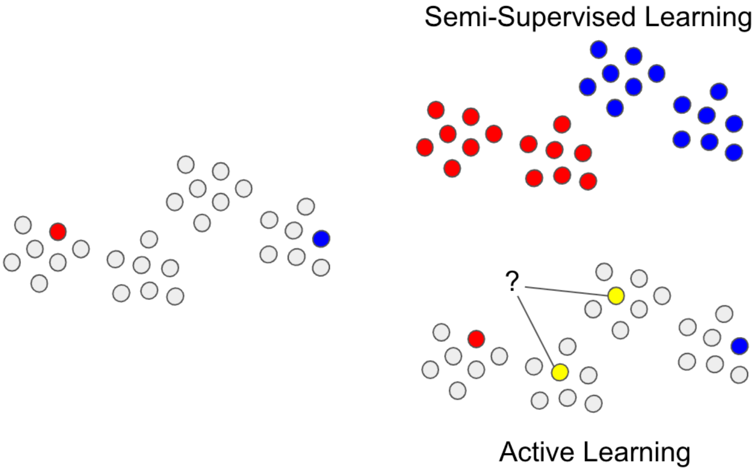
Semi-Supervised Learning

From the given data, can we accurately infer the labelings on $\mathcal{U}$?

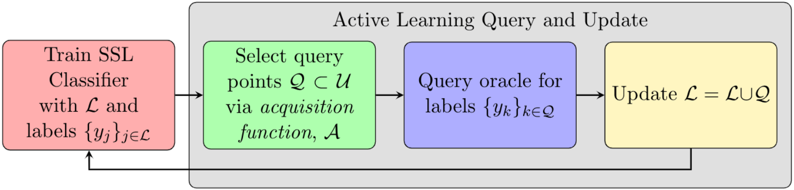
Active Learning

From the given data, can we judiciously "choose" unlabeled points $\mathcal{Q} \subset \mathcal{U}$ to label that will improve the output of the underlying learning model?

Active Learning vs Semi-Supervised Learning



Active Learning



Graph Construction

Given data $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, construct *similarity graph* $G(Z, W)$, where

- $Z = \{1, 2, \ldots, N\}$
- $W_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$
- $d_i = \sum_{j \in Z} W_{ij}$
- degree matrix $D = \operatorname{diag}(d_1, d_2, \ldots, d_N)$

Graph Laplacians

- $L = D - W$, *unnormalized*
- $L_n = I - D^{-1/2} W D^{-1/2}$, *normalized*
- $L_{rw} = I - D^{-1} W$, *random walk*

Binary Graph-Based SSL

Consider family of graph-based SSL models, using a perturbed *graph Laplacian* $L_\tau = L + \tau^2 I$:

$$\hat{\mathbf{u}} = \operatorname*{arg\,min}_{\mathbf{u}\in\mathbb{R}^N} \frac{1}{2}\langle \mathbf{u}, L_\tau \mathbf{u}\rangle + \sum_{j\in\mathcal{L}} \ell(u_j, y_j) =: \operatorname*{arg\,min}_{\mathbf{u}\in\mathbb{R}^N} J_\ell(\mathbf{u};\mathbf{y}), \qquad (1)$$

for different loss functions ℓ with parameter γ:

- $\ell(x,y) = (x-y)^2/2\gamma^2$, (Regression)
- $\ell(x,y) = \ln(1+e^{-xy/\gamma})$, (Logistic)
- $\ell(x,y) = -\ln \Psi_\gamma(xy)$, (Probit)

where $\Psi_\gamma(t) = \int_{-\infty}^t \psi_\gamma(s)ds$ is CDF of log-concave PDF $\psi_\gamma(s)$.

Multiclass Graph-Based SSL

With perturbed graph Laplacian L_τ and n_c the number of classes,

$$\hat{U} = \underset{U \in \mathbb{R}^{N \times n_c}}{\arg\min} \ \frac{1}{2}\langle U, L_\tau U \rangle_F + \sum_{j \in \mathcal{L}} \ell(\mathbf{u}^j, \mathbf{y}^j) =: \underset{U \in \mathbb{R}^{N \times n_c}}{\arg\min} \ \mathcal{J}_\ell(U; Y),$$

for different loss functions ℓ with parameter γ:

- $\ell(\mathbf{s}, \mathbf{t}) = \frac{1}{2\gamma^2}\|\mathbf{s} - \mathbf{t}\|_2^2$, (Multiclass Regression)
- $\ell(\mathbf{s}, \mathbf{t}) = -\sum_{c=1}^{n_c} t_c \ln(s_c)$, (Cross-Entropy)

Probabilistic and Bayesian Perspective

Optimizer $\hat{\mathbf{u}}$ can be viewed as *maximum a posteriori* (MAP) estimator

$$\begin{aligned}\arg\min_{\mathbf{u}} J_\ell(\mathbf{u};\mathbf{y}) \iff \arg\max_{\mathbf{u}} \; \exp(-J_\ell(\mathbf{u};\mathbf{y})) \\ = \arg\max_{\mathbf{u}} \; \underbrace{\exp\left(-\frac{1}{2}\langle \mathbf{u}, L_\tau \mathbf{u}\rangle\right)}_{prior} \underbrace{\exp\left(-\sum_{j\in\mathcal{L}} \ell(u_j, y_j)\right)}_{likelihood} \\ = \arg\max_{\mathbf{u}} \; \mathbb{P}(\mathbf{u}|\mathbf{y})\end{aligned}$$

for a posterior distribution $\mathbb{P}(\mathbf{u}|\mathbf{y}) \propto \exp(-J_\ell(\mathbf{u};\mathbf{y}))$.

- Different loss functions give different likelihoods

Side Note

Ginzburg-Landau/Graph MBO?

$$J(\mathbf{u};\mathbf{y}) = \frac{1}{2}\langle \mathbf{u}, L_\tau \mathbf{u}\rangle + \overbrace{\frac{1}{4\epsilon}\sum_{i\in Z}(u_i^2-1)^2}^{\text{double-well potential}} + \frac{\lambda}{2}\sum_{j\in\mathcal{L}}(u_j - y_j)^2$$

- **non-convex**
- corresponding posterior?

$$\mathbb{P}(\mathbf{u}|\mathbf{y}) \propto \exp\left(\frac{-1}{2}\langle \mathbf{u}, L_\tau \mathbf{u}\rangle + \frac{1}{4\epsilon}\sum_{i\in Z}(u_i^2-1)^2\right)\exp\left(\frac{-\lambda}{2}\sum_{j\in\mathcal{L}}(u_j - y_j)^2\right)$$

- non-Gaussian prior, Gaussian likelihood
- multimodal distribution

Harmonic Functions (HF) Model

Assuming hard constraints for labeling[1], have conditional distribution:

$$\mathbf{u}_{\mathcal{U}}|\mathbf{y} \sim \mathcal{N}(\mathbf{u}_{hf}, L_{\mathcal{U},\mathcal{U}}^{-1}), \quad \mathbf{u}_{hf} = -L_{\mathcal{U},\mathcal{U}}^{-1} L_{\mathcal{U},\mathcal{L}} \mathbf{y}$$

with $\mathbf{u}_{\mathcal{L}} = \mathbf{y}$.

Gaussian Regression (GR) Model

With $\ell(x, y) = (x - y)^2/2\gamma^2$, then likelihood/prior/posterior is Gaussian.

$$\mathbb{P}(\mathbf{u}|\mathbf{y}) \propto \exp\left(-\frac{1}{2}\langle \mathbf{u}, L_\tau \mathbf{u}\rangle\right) \exp\left(-\frac{1}{2\gamma^2}\sum_{j\in\mathcal{L}}(u_j - y_j)^2\right)$$

$$\sim \mathcal{N}(\mathbf{m}, C), \ \mathbf{m} = \frac{1}{\gamma^2} C P^T \mathbf{y}, \ C^{-1} = L + \frac{1}{\gamma^2} P^T P,$$

where $P : \mathbb{R}^N \to \mathbb{R}^{|\mathcal{L}|}$ is projection onto labeled set $\mathcal{L}$.

[1] Does not actually rigorously fit into Bayesian framework like others

Look-Ahead model with index k and label y_k:

$$\arg\min_{\mathbf{u}\in\mathbb{R}^N} J^k(\mathbf{u};\mathbf{y},y_k) := \arg\min_{\mathbf{u}\in\mathbb{R}^N} \ \frac{1}{2}\langle \mathbf{u}, L_\tau \mathbf{u}\rangle + \sum_{j\in\mathcal{L}} \ell(u_j,y_j) + \overbrace{\ell(u_k,y_k)}^{plus\ \ k}.$$

- For Gaussian model, look-ahead posterior distribution's parameters from the current posterior distribution
  - *without expensive model retraining* – **rank-one updates**

GR: $\mathbf{m}^{k,y_k} = \mathbf{m} + \frac{(y_k - m_k)}{\gamma^2 + C_{kk}} C_{:,k}, \quad C^{k,y_k} = C - \frac{1}{\gamma^2 + C_{kk}} C_{:,k}C_{:,k}^T$

Non-Gaussian, Probit

When likelihood not Gaussian, posterior $\mathbb{P}(\mathbf{u}|\mathbf{y})$ *is non-Gaussian..*

Problems:

- model classifier as mean $\mu = \mathbb{E}_{\mathbf{u}\sim\mathbb{P}}[\mathbf{u}]$? or MAP estimator $\hat{\mathbf{u}} = \arg\max \mathbb{P}(\mathbf{u}|\mathbf{y})$?
- compute mean, μ, and covariance $C = \mathbb{E}_{\mathbf{u}\sim\mathbb{P}}\left[(\mathbf{u}-\mu)(\mathbf{u}-\mu)^T\right]$? (*potentially expensive!*)
- Look-ahead updates??

With non-Gaussian models, we lose these nice properties. *What to do?*

Non-Gaussian, Probit

When likelihood not Gaussian, posterior $\mathbb{P}(\mathbf{u}|\mathbf{y})$ *is non-Gaussian..*

Problems:

- model classifier as mean $\mu = \mathbb{E}_{\mathbf{u}\sim\mathbb{P}}[\mathbf{u}]$? or MAP estimator $\hat{\mathbf{u}} = \arg\max \mathbb{P}(\mathbf{u}|\mathbf{y})$?
- compute mean, μ, and covariance $C = \mathbb{E}_{\mathbf{u}\sim\mathbb{P}}\left[(\mathbf{u}-\mu)(\mathbf{u}-\mu)^T\right]$? (*potentially expensive!*)
- Look-ahead updates??

With non-Gaussian models, we lose these nice properties. *What to do?*

Let's approximate with Gaussian, and see what happens!

Laplace approximation is a popular technique for approximating non-Gaussian distributions $\mathbb{P}$ with a Gaussian distribution.

$$\mathbf{x} \sim \mathcal{N}(\hat{\mathbf{x}}, \hat{C}), \quad \hat{\mathbf{x}} = \underset{\mathbf{x}\in\mathbb{R}^N}{\arg\max}\ \mathbb{P}(\mathbf{x}), \quad \hat{C} = \left(-\nabla^2 \ln(\mathbb{P}(\mathbf{x}))|_{\mathbf{x}=\hat{\mathbf{x}}}\right)^{-1},$$

where

- $\hat{\mathbf{x}}$: MAP estimator of $\mathbb{P}$
- $\hat{C}$: Hessian matrix of the negative-log density of $\mathbb{P}$, evaluated at $\hat{\mathbf{x}}$

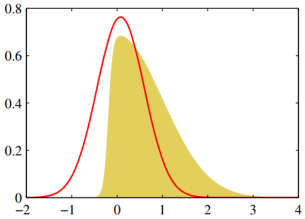


Figure 1: photo credit : http://wiljohn.top/2019/04/14/PRML4-4/

Consider only first $M < N$ eigenvalues and eigenvectors of graph Laplacian, L:

$$0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_M, \quad \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_M.$$

- $\Lambda_\tau = \text{diag}\left(\lambda_1 + \tau^2, \ldots, \lambda_M + \tau^2\right)$
- $V = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \ldots \quad \mathbf{v}_M] \in \mathbb{R}^{N \times M}$
- $\boldsymbol{\alpha} \in \mathbb{R}^M$ (binary), $A \in \mathbb{R}^{M \times n_c}$ (multiclass)

Spectral Truncation

Consider only first $M < N$ eigenvalues and eigenvectors of graph Laplacian, L:

$$0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_M, \quad \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_M.$$

- $\Lambda_\tau = \mathrm{diag}\left(\lambda_1 + \tau^2, \ldots, \lambda_M + \tau^2\right)$
- $V = [\mathbf{v}_1 \;\; \mathbf{v}_2 \;\; \ldots \;\; \mathbf{v}_M] \in \mathbb{R}^{N \times M}$
- $\boldsymbol{\alpha} \in \mathbb{R}^M$ (binary), $A \in \mathbb{R}^{M \times n_c}$ (multiclass)

Binary: $(\mathbf{u} = V\boldsymbol{\alpha})$

$$J_\ell(\mathbf{u}; \mathbf{y}) = \frac{1}{2}\langle \mathbf{u}, L_\tau \mathbf{u} \rangle + \sum_{j \in \mathcal{L}} \ell(u_j, y_j)$$

$$\rightarrow \frac{1}{2}\langle \boldsymbol{\alpha}, \Lambda_\tau \boldsymbol{\alpha} \rangle + \sum_{j \in \mathcal{L}} \ell(\mathbf{e}_j^T V \boldsymbol{\alpha}, y_j) =: \tilde{J}_\ell(\boldsymbol{\alpha}; \mathbf{y}),$$

Consider only first $M < N$ eigenvalues and eigenvectors of graph Laplacian, L:

$$0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_M, \quad \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_M.$$

- $\Lambda_\tau = \text{diag}\left(\lambda_1 + \tau^2, \ldots, \lambda_M + \tau^2\right)$
- $V = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \ldots \quad \mathbf{v}_M] \in \mathbb{R}^{N \times M}$
- $\boldsymbol{\alpha} \in \mathbb{R}^M$ (binary), $A \in \mathbb{R}^{M \times n_c}$ (multiclass)

Binary: $(\mathbf{u} = V\boldsymbol{\alpha})$

$$J_\ell(\mathbf{u}; \mathbf{y}) = \frac{1}{2}\langle \mathbf{u}, L_\tau \mathbf{u}\rangle + \sum_{j \in \mathcal{L}} \ell(u_j, y_j)$$

$$\to \frac{1}{2}\langle \boldsymbol{\alpha}, \Lambda_\tau \boldsymbol{\alpha}\rangle + \sum_{j \in \mathcal{L}} \ell(\mathbf{e}_j^T V \boldsymbol{\alpha}, y_j) =: \tilde{J}_\ell(\boldsymbol{\alpha}; \mathbf{y}),$$

Multiclass: $(U = VA)$

$$\mathcal{J}_\ell(U; Y) = \frac{1}{2}\langle U, L_\tau U\rangle_F + \sum_{j \in \mathcal{L}} \ell(\mathbf{u}^j, \mathbf{y}^j)$$

$$\to \frac{1}{2}\langle A, \Lambda_\tau A\rangle_F + \sum_{j \in \mathcal{L}} \ell(\mathbf{e}_j^T V A, \mathbf{y}^j) =: \tilde{\mathcal{J}}_\ell(A; Y).$$

$$\boldsymbol{\alpha}|\mathbf{y} \sim \mathcal{N}(\hat{\boldsymbol{\alpha}}, \hat{C}_{\hat{\boldsymbol{\alpha}}}), \qquad \hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}\in\mathbb{R}^M}{\arg\min}\ \tilde{J}_\ell(\boldsymbol{\alpha};\mathbf{y}),$$

and then calculate covariance of Laplace Approximation $\hat{C}_{\boldsymbol{\alpha}}$

$$\nabla_{\boldsymbol{\alpha}} \tilde{J}_\ell(\boldsymbol{\alpha};\mathbf{y}) = \Lambda_\tau \boldsymbol{\alpha} + \sum_{j\in\mathcal{L}} F(\mathbf{e}_j^T V\boldsymbol{\alpha}, y_j) V^T \mathbf{e}_j = \Lambda_\tau \boldsymbol{\alpha} + V^T \sum_{j\in\mathcal{L}} F(\mathbf{e}_j^T V\boldsymbol{\alpha}, y_j)\mathbf{e}_j,$$

$$\nabla^2_{\boldsymbol{\alpha}} \tilde{J}_\ell(\boldsymbol{\alpha};\mathbf{y}) = \Lambda_\tau + V^T \left(\sum_{j\in\mathcal{L}} F'(\mathbf{e}_j^T V\boldsymbol{\alpha}, y_j)\mathbf{e}_j\mathbf{e}_j^T \right) V,$$

$$\implies \hat{C}_{\boldsymbol{\alpha}} = \left(\nabla^2_{\boldsymbol{\alpha}} \tilde{J}_\ell(\boldsymbol{\alpha};\mathbf{y})\right)^{-1} = \left(\Lambda_\tau + V^T \left(\sum_{j\in\mathcal{L}} F'(\mathbf{e}_j^T V\boldsymbol{\alpha}, y_j)\mathbf{e}_j\mathbf{e}_j^T \right) V \right)^{-1}$$

How to approximate look-ahead model update, $\hat{\boldsymbol{\alpha}}^{k,y_k} = \arg\min \tilde{J}_\ell^{k,y_k}$?

- have $\hat{C}_{\hat{\boldsymbol{\alpha}}}$ (i.e. *inverse Hessian*)

How to approximate look-ahead model update, $\hat{\boldsymbol{\alpha}}^{k,y_k} = \arg\min \tilde{J}_\ell^{k,y_k}$?

- have $\hat{C}_{\hat{\boldsymbol{\alpha}}}$ (i.e. *inverse Hessian*)

Try one step of Newton's method, *starting at* $\hat{\boldsymbol{\alpha}}$:

$$\begin{aligned}
\tilde{\boldsymbol{\alpha}}^{k,y_k} &= \hat{\boldsymbol{\alpha}} - \left(\nabla^2_{\boldsymbol{\alpha}} \tilde{J}_\ell^{k,y_k}(\hat{\boldsymbol{\alpha}}; \mathbf{y}, y_k)\right)^{-1} \left(\nabla_{\boldsymbol{\alpha}} \tilde{J}_\ell^{k,y_k}(\hat{\boldsymbol{\alpha}}; \mathbf{y}, y_k)\right) \\
&= \ldots \\
&= \hat{\boldsymbol{\alpha}} - \frac{F((\mathbf{v}^k)^T \hat{\boldsymbol{\alpha}}, y_k)}{1 + F'((\mathbf{v}^k)^T \boldsymbol{\alpha}, y_k)(\mathbf{v}^k)^T \hat{C}_{\hat{\boldsymbol{\alpha}}} \mathbf{v}^k} \hat{C}_{\hat{\boldsymbol{\alpha}}} \mathbf{v}^k
\end{aligned}$$

where

$$F(x,y) := \frac{\partial \ell}{\partial x}(x,y), \; F'(x,y) := \frac{\partial^2 \ell}{\partial x^2}(x,y).$$

How to approximate look-ahead model update, $\hat{\boldsymbol{\alpha}}^{k,y_k} = \arg\min \tilde{J}_\ell^{k,y_k}$?

- have $\hat{C}_{\hat{\boldsymbol{\alpha}}}$ (i.e. *inverse Hessian*)

Try one step of Newton's method, *starting at* $\hat{\boldsymbol{\alpha}}$:

$$\begin{aligned}\tilde{\boldsymbol{\alpha}}^{k,y_k} &= \hat{\boldsymbol{\alpha}} - \left(\nabla^2_{\boldsymbol{\alpha}} \tilde{J}_\ell^{k,y_k}(\hat{\boldsymbol{\alpha}}; \mathbf{y}, y_k)\right)^{-1} \left(\nabla_{\boldsymbol{\alpha}} \tilde{J}_\ell^{k,y_k}(\hat{\boldsymbol{\alpha}}; \mathbf{y}, y_k)\right) \\ &= \dots \\ &= \hat{\boldsymbol{\alpha}} - \frac{F((\mathbf{v}^k)^T \hat{\boldsymbol{\alpha}}, y_k)}{1 + F'((\mathbf{v}^k)^T \boldsymbol{\alpha}, y_k)(\mathbf{v}^k)^T \hat{C}_{\hat{\boldsymbol{\alpha}}} \mathbf{v}^k} \hat{C}_{\hat{\boldsymbol{\alpha}}} \mathbf{v}^k\end{aligned}$$

where

$$F(x,y) := \frac{\partial \ell}{\partial x}(x,y), \;\; F'(x,y) := \frac{\partial^2 \ell}{\partial x^2}(x,y).$$

Simple update!

$*$ GR: this reduces to the exact look-ahead update!

Approximate Look-Ahead - Multiclass

Similar result for multiclass case, but a little lengthy to describe...

Approximate Look-Ahead - Multiclass

Similar result for multiclass case, but a little lengthy to describe...

$$\tilde{A}^{k,y_k} = \hat{A} - \underbrace{\left(\nabla_A^2 \tilde{\mathcal{J}}^{k,y_k}(\hat{A}; Y, \mathbf{y}^k)\right)^{-1} \left(\nabla_A \tilde{\mathcal{J}}^{k,y_k}(\hat{A}; Y, \mathbf{y}^k)\right)}_{\text{simplifies to be rank } n_c}$$

Model Change Acquisition Function

Calculating the approximate change in a model (i.e. classifier) from the addition of an index k and associated label y_k has been investigated previously[2].

[2] Cai, Zhang, and Zhou, "Maximizing Expected Model Change for Active Learning in Regression", 2013; Karzand and Nowak, "MaxiMin Active Learning in Overparameterized Model Classes", 2020.

Model Change Acquisition Function

Calculating the approximate change in a model (i.e. classifier) from the addition of an index k and associated label y_k has been investigated previously[2].

Employ approximate update (recalling that $V\boldsymbol{\alpha} = \mathbf{u}$):

$$\begin{aligned}
\mathcal{A}(k) &= \min_{y_k \in \{\pm 1\}} \|\hat{\mathbf{u}}^{k,y_k} - \hat{\mathbf{u}}\|_2 \approx \min_{y_k \in \{\pm 1\}} \left\|\tilde{\mathbf{u}}^{k,y_k} - \hat{\mathbf{u}}\right\|_2 = \min_{y_k \in \{\pm 1\}} \left\|\tilde{\boldsymbol{\alpha}}^{k,y_k} - \hat{\boldsymbol{\alpha}}\right\|_2 \\
&= \min_{y_k \in \{\pm 1\}} \left| \frac{F((\mathbf{v}^k)^T \hat{\boldsymbol{\alpha}}, y_k)}{1 + F'((\mathbf{v}^k)^T \hat{\boldsymbol{\alpha}}, y_k)(\mathbf{v}^k)^T \hat{C}_{\hat{\boldsymbol{\alpha}}} \mathbf{v}^k} \right| \left\|\hat{C}_{\hat{\boldsymbol{\alpha}}} \mathbf{v}^k\right\|_2 \\
&= \min_{y_k \in \{\pm 1\}} \left| \frac{F(\hat{u}_k, y_k)}{1 + F'(\hat{u}_k, y_k)(\mathbf{v}^k)^T \hat{C}_{\hat{\boldsymbol{\alpha}}} \mathbf{v}^k} \right| \left\|\hat{C}_{\hat{\boldsymbol{\alpha}}} \mathbf{v}^k\right\|_2,
\end{aligned}$$

[2] Cai, Zhang, and Zhou, "Maximizing Expected Model Change for Active Learning in Regression", 2013; Karzand and Nowak, "MaxiMin Active Learning in Overparameterized Model Classes", 2020.

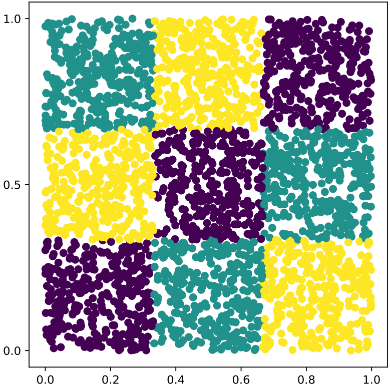
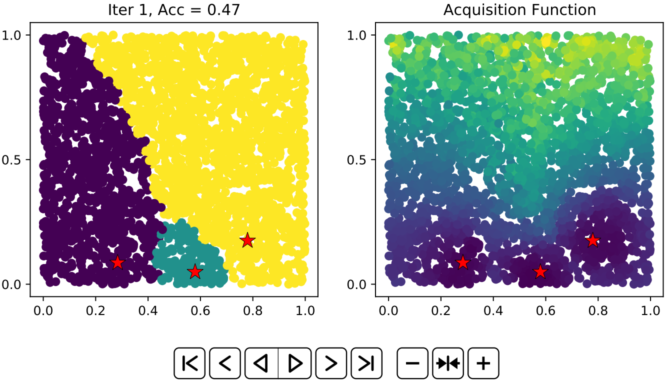


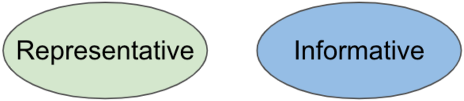
Figure 2: Checkerboard 3 Dataset Ground Truth

Demo



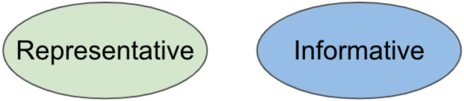
Connection to Reinforcement Learning

Active Learning – select ‘useful” points to label that will improve your classifier



- **Representative** : “looks” representative of the data
- **Informative** : help to refine the classifier’s decision boundary

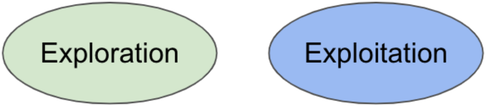
Active Learning – select ‘useful” points to label that will improve your classifier



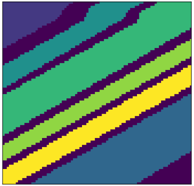
- **Representative** : “looks” representative of the data
- **Informative** : help to refine the classifier’s decision boundary

---

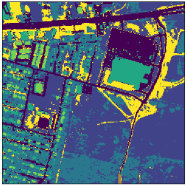
Reinforcement Learning – learn optimal policy via sequential decision making



- **Exploration** : “explore” the inherent geometric/clustering structure
- **Exploitation** : “exploit” the classification structure that have learned so far



(a) Salinas A



(b) Urban

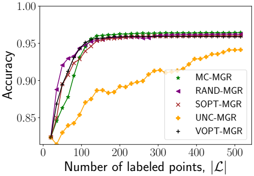
Graph Construction:

- 15 nearest neighbors, cosine similarity
- Zelnik-Perona scaling
- $M = 50$ eigenvalues

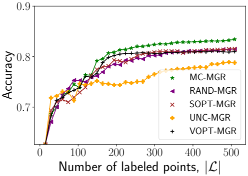
Experiments:

- initially label 2 per class
- Batch
  - 100 active learning iterations, select $B = 5$ query points at each iteration
  - MGR (Multiclass Gaussian Regression)
  - CE (Cross-Entropy)

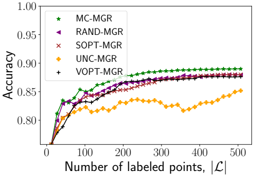
Multiclass GR Results:



(c) MNIST

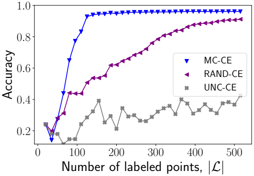


(d) Salinas A

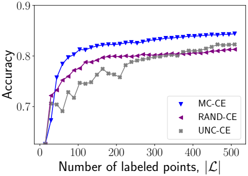


(e) Urban

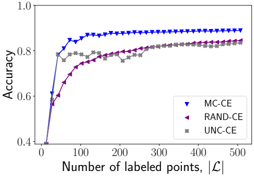
Cross-Entropy Results:



(f) MNIST



(g) Salinas A



(h) Urban

Future Directions

- Adapt this to more useful GBSSL models?
  - Currently only viable for convex loss functions (i.e. Laplace Approximation)
  - e.g. graph MBO posterior is multimodal, so Laplace approximation meaningful?
- Other active learning criterion that take advantage of the nice model properties we have here?
- Deep Learning?

Why not apply this work?

[3] Cai, Zhang, and Zhou, "Maximizing Expected Model Change for Active Learning in Regression", 2013.

[4] Ash et al., "Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds", 2020.

[5] Gal, Islam, and Ghahramani, "Deep Bayesian active learning with image data", 2017.

Why not apply this work?

Neural network $F_\theta(\cdot)$, parameterized by weights $\theta \in \mathbb{R}^D$ (D usually **very large**).

$$J(\theta; \mathcal{X}, \mathbf{y}) = \sum_{i=1}^{N} \ell(F_\theta(\mathbf{x}_i), y_i) + \mathcal{R}(\theta)$$

- supervised vs semi-supervised learning

[3] Cai, Zhang, and Zhou, "Maximizing Expected Model Change for Active Learning in Regression", 2013.

[4] Ash et al., "Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds", 2020.

[5] Gal, Islam, and Ghahramani, "Deep Bayesian active learning with image data", 2017.

Why not apply this work?

Neural network $F_\theta(\cdot)$, parameterized by weights $\theta \in \mathbb{R}^D$ (D usually **very large**).

$$J(\theta; \mathcal{X}, \mathbf{y}) = \sum_{i=1}^{N} \ell(F_\theta(\mathbf{x}_i), y_i) + \mathcal{R}(\theta)$$

- supervised vs semi-supervised learning
- look-ahead? model change?
  - inverse Hessian – $\mathcal{O}(D^2)$ for NN :(
  - approximate model change via approximated gradient $\frac{\partial J}{\partial \theta}$ (Cai et al[3])
  - cluster on space of gradients[4]

[3] Cai, Zhang, and Zhou, "Maximizing Expected Model Change for Active Learning in Regression", 2013.

[4] Ash et al., "Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds", 2020.

[5] Gal, Islam, and Ghahramani, "Deep Bayesian active learning with image data", 2017.

Why not apply this work?

Neural network $F_\theta(\cdot)$, parameterized by weights $\theta \in \mathbb{R}^D$ (D usually **very large**).

$$J(\theta; \mathcal{X}, \mathbf{y}) = \sum_{i=1}^{N} \ell(F_\theta(\mathbf{x}_i), y_i) + \mathcal{R}(\theta)$$

- supervised vs semi-supervised learning
- look-ahead? model change?
  - inverse Hessian – $\mathcal{O}(D^2)$ for NN :(
  - approximate model change via approximated gradient $\frac{\partial J}{\partial \theta}$ (Cai et al[3])
  - cluster on space of gradients[4]
- Bayesian interpretation?
  - F_θ non-linear, J highly non-convex –> multimodal distribution
  - MCMC-“esque” sampling from posterior via Dropout[5]

[3] Cai, Zhang, and Zhou, “Maximizing Expected Model Change for Active Learning in Regression”, 2013.

[4] Ash et al., “Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds”, 2020.

[5] Gal, Islam, and Ghahramani, “Deep Bayesian active learning with image data”, 2017.

Overview

Overall Advice

- Cultivate relationships with multiple professors/mentors
- Resume
  - Research experience (e.g. REU, undergraduate research)
  - Math Subject GRE
- Research Statement + Personal Statement – do your homework
- Big vs Small & "Traditional" vs "Newer" Programs

Ph.D. Program Timeline

Qualifying Exams

(Years 1-2)

Align with Advisor

(Year ~2.5)

Publish Papers / Finish Dissertation

(Years 3-5)


Jump Through Hoops
xams
(Years 1-2)
“Dating” Period
Advisor
(Year ~2.5)
The Marathon
pers /
Dissertation
(Years 3-5)

Ash, Jordan T. et al. "Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=ryghZJBKPS.

Cai, Wenbin, Ya Zhang, and Jun Zhou. "Maximizing Expected Model Change for Active Learning in Regression". In: *2013 IEEE 13th International Conference on Data Mining*. ISSN: 2374-8486. Dec. 2013, pp. 51–60. DOI: 10.1109/ICDM.2013.104.

Gal, Yarin, Riashat Islam, and Zoubin Ghahramani. "Deep Bayesian active learning with image data". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, Aug. 2017, pp. 1183–1192. (Visited on 06/11/2020).

Karzand, Mina and Robert D. Nowak. "MaxiMin Active Learning in Overparameterized Model Classes". In: *IEEE Journal on Selected Areas in Information Theory* 1.1 (May 2020). Conference Name: IEEE Journal on Selected Areas in Information Theory, pp. 167–177. ISSN: 2641-8770. DOI: 10.1109/JSAIT.2020.2991518.

Miller, Kevin, Hao Li, and Andrea L Bertozzi. "Efficient Graph-Based Active Learning with Probit Likelihood via Gaussian Approximations". en. In: *ICML* Workshop on Real-World Experiment Design and Active Learning (2020).

Rasmussen, Carl Edward and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. Cambridge, Mass: MIT Press, 2006. ISBN: 978-0-262-18253-9.

Settles, Burr. "Active Learning". en. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6.1 (June 2012), pp. 1–114. ISSN: 1939-4608, 1939-4616. DOI: 10.2200/S00429ED1V01Y201207AIM018. URL: http://www.morganclaypool.com/doi/abs/10.2200/S00429ED1V01Y201207AIM018 (visited on 06/11/2020).

Zhu, Xiaojin, Zoubin Ghahramani, and John Lafferty. "Semi-supervised learning using Gaussian fields and harmonic functions". In: *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*. ICML'03. Washington, DC, USA: AAAI Press, Aug. 2003, pp. 912–919. ISBN: 978-1-57735-189-4. (Visited on 06/11/2020).

Zhu, Xiaojin, John Lafferty, and Zoubin Ghahramani. "Combining Active Learning and Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions". In: *ICML 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*. 2003, pp. 58–65.