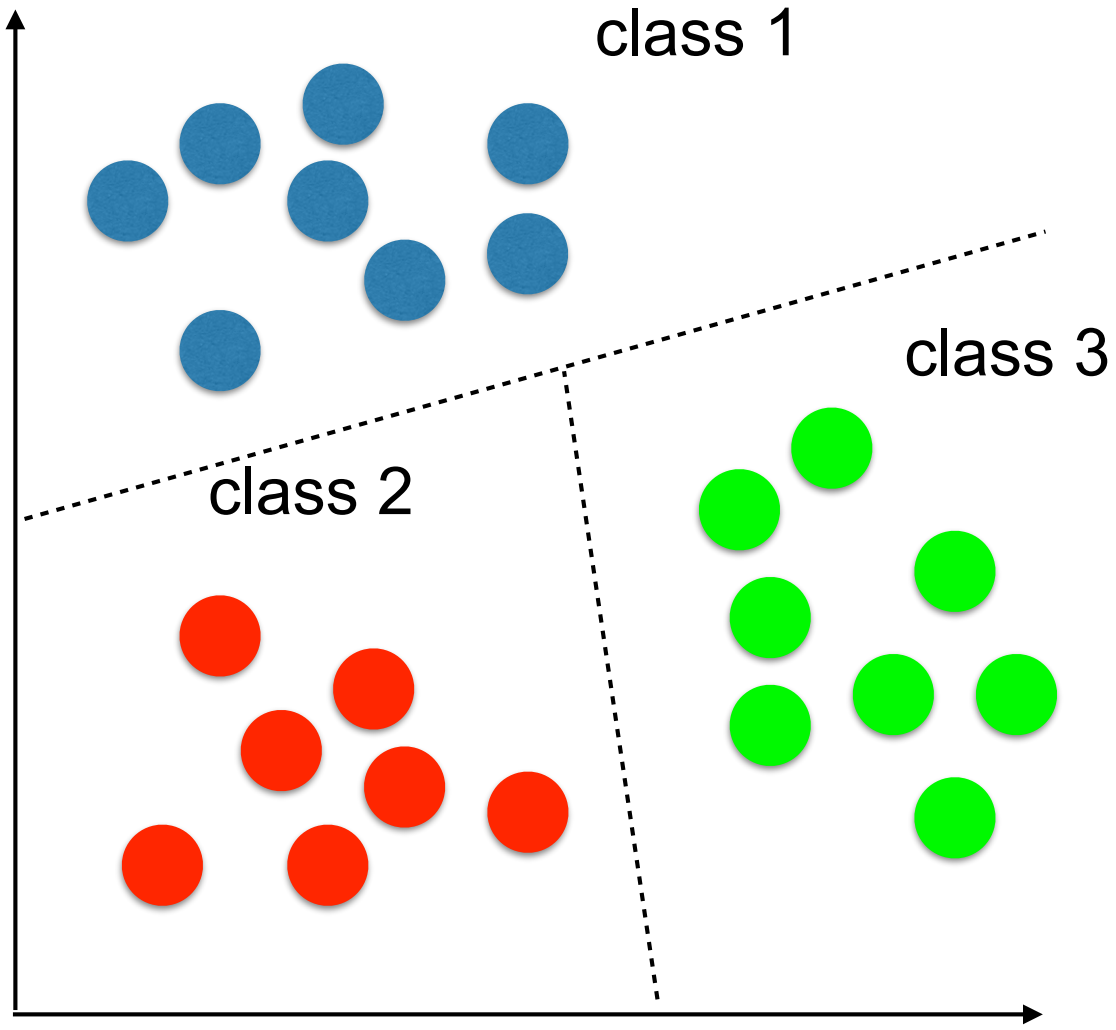
Accelerating Cross-Validation in Multinomial Logistic Regression with ℓ_1-Regularization

Tomoyuki Obuchi
Dep. Math. Comp. Sci.
Tokyo Tech.

In Collaboration with Yoshiyuki Kabashima in Tokyo Tech.

Multi-class classification

- Under supervised setting



Multinomial Logistic Regression (MLR)

- Multinomial logistic regression (called also as softmax function)

$$P(y=a|\boldsymbol{x};\{\boldsymbol{w}_a\}_a)=\frac{e^{\boldsymbol{x}\cdot\boldsymbol{w}_a}}{\sum_{b=1}^{L}e^{\boldsymbol{x}\cdot\boldsymbol{w}_b}}$$

- Teacher Data: $D^M=\{(\boldsymbol{x}_\mu,y_\mu)\}_{\mu=1}^M$
  - Feature vector: $\boldsymbol{x}\in\mathbb{R}^N$
  - Class variable: $y\in\{1,2,\cdots,L\}$

- A common learning method: Maximum likelihood (ML) approach

$$\{\hat{\boldsymbol{w}}_a\}_a=\underset{\{\boldsymbol{w}_a\}_a}{\arg\min}\left\{-\sum_{\mu=1}^{M}\log\phi\left(y_\mu|\{u_{\mu b}=\boldsymbol{x}_\mu^\top\boldsymbol{w}_b\}_b\right)\right\}$$

$\{\boldsymbol{w}_a\in\mathbb{R}^N\}_{a=1}^L$: Weight or Coefficient

Regularization

- ML approach can be unstable
  - If the dataset size is not large enough
  - The data generating process largely deviates from the MLR

←Regularization technique

- Penalized ML approach

$$\{\hat{\boldsymbol{w}}_a(\boldsymbol{\lambda})\}_a = \min_{\{\boldsymbol{w}_a\}_a} \left\{ \sum_{\mu=1}^{M} q_\mu \left(\{\boldsymbol{w}_a\}_{a=1}^{L} \right) + J(\{\boldsymbol{w}\}_{a=1}^{L} | \boldsymbol{\lambda}) \right\}$$

  - Negative log-likelihood

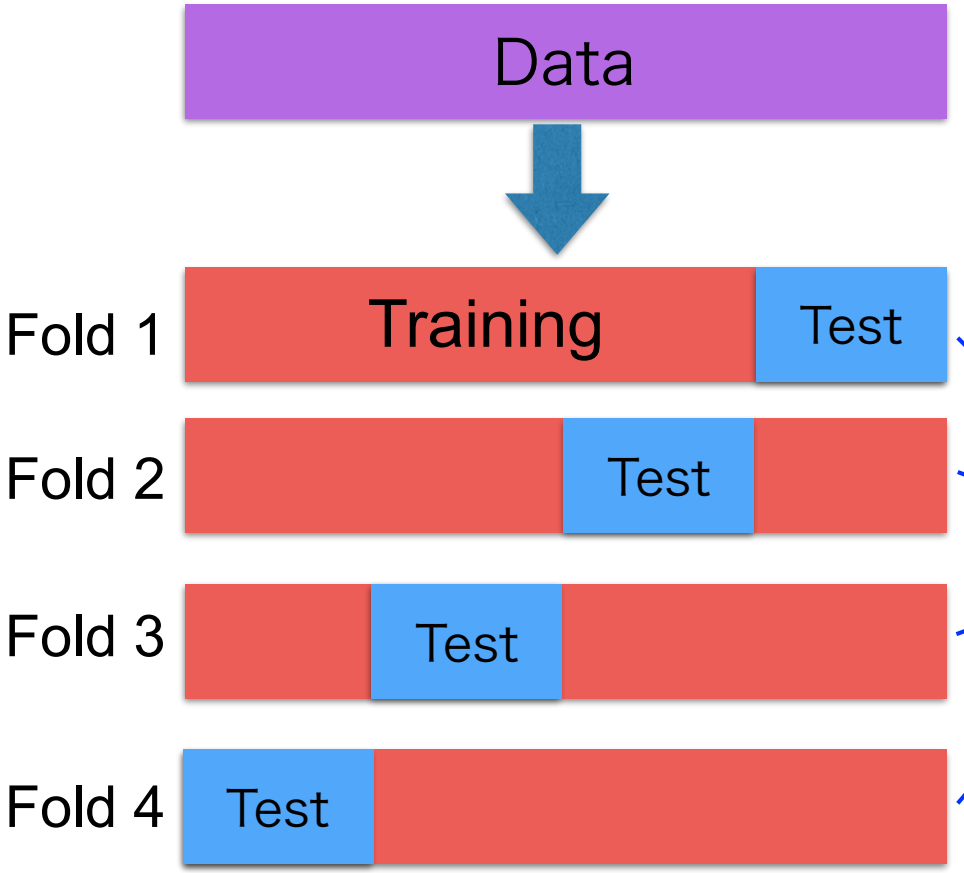
$$q_\mu \left(\{\boldsymbol{w}_a\}_{a=1}^{L} \right) = -\ln \phi \left(y_\mu \middle| \left\{ u_{\mu a} = \boldsymbol{x}_\mu^\top \boldsymbol{w}_a \right\}_{a=1}^{L} \right),$$

- How to determine $\boldsymbol{\lambda}$? ←Examine generalization error

→ Cross-Validation (CV)

Cross-validation (CV)

- Dividing dataset into training set and test set
  - Learning on Training Set
  - Examine generalization error on Test Set



k-fold CV: The original sample is randomly partitioned into k equal sized subsamples

(left panel is for k=4)

Average

CV Error= An estimator of generalization error

LOOCV and Linear Response

- Leave-one-out (LOO) CV (=M-fold CV), Predictive likelihood

$$\{\hat{\boldsymbol{w}}_a^{\setminus\mu}(\lambda)\}_a = \underset{\{\boldsymbol{w}_a\}_a}{\arg\min} \left\{ \mathcal{H}\left(\{\boldsymbol{w}_a\}_{a=1}^{L} \middle| D^M, \lambda \right) - q_\mu\left(\{\boldsymbol{w}_a\}_{a=1}^{L} \right) \right\}$$

$$\epsilon_{\mathrm{LOO}}(\lambda) = \frac{1}{M} \sum_{\mu=1}^{M} q_\mu \left(\left\{ \hat{\boldsymbol{w}}_a^{\setminus\mu} \right\}_{a=1}^{L} \right)$$

←Computationally expensive
(*M* times repeated optimization)

- Diff. between Full and LOO solutions owes to only one term of q_μ → Linear (response) approximation!

- Approximate formula (called ACV formula):

$$\hat{\boldsymbol{u}}_\mu^{\setminus\mu} \approx \hat{\boldsymbol{u}}_\mu + C_\mu \left(I - C_\mu F_\mu \right)^{-1} \boldsymbol{b}_\mu$$

  - Crucial assumption for the formula: The set of active variables is common between full and LOO systems.

Ingredients in approximation

$$\hat{\boldsymbol{u}}_\mu^{\backslash\mu} \approx \hat{\boldsymbol{u}}_\mu + C_\mu \left(I - C_\mu F_\mu\right)^{-1} \boldsymbol{b}_\mu$$

$$C_\mu = X^\mu_{*\hat{A}} \left(G_{\hat{A}\hat{A}}\right)^{-1} \left(X^\mu_{*\hat{A}}\right)^\top \qquad G_{(ai)(bj)} \equiv \frac{\partial^2 \mathcal{H}}{\partial w_{ai} \partial w_{bj}} \equiv \left(\partial^2 \mathcal{H}\right)_{(ai)(bj)}$$

$$\hat{A} = \{(ai) | \hat{w}_{ai} \neq 0\}$$

$$X^\mu = \underbrace{\begin{pmatrix} \boxed{\boldsymbol{x}_\mu^\top} & & 0 \\ & \ddots & \\ 0 & & \boxed{\boldsymbol{x}_\mu^\top} \end{pmatrix}}_{NL} \Bigg\} L$$

$$F^\mu_{ab} \equiv \frac{\partial^2 q_\mu}{\partial u_{\mu a} \partial u_{\mu b}} = \delta_{ab} p_{a|\mu} - p_{a|\mu} p_{b|\mu} \qquad p_{a|\mu} = \phi(a | \{\hat{u}_{\mu b}\}_b)$$

$$\boldsymbol{b}_\mu = \frac{\partial q_\mu}{\partial \boldsymbol{u}_\mu} = \left(p_{1|\mu} - \delta_{1 y_\mu}, p_{2|\mu} - \delta_{2 y_\mu}, \cdots, p_{L|\mu} - \delta_{L y_\mu}\right)^\top$$

Derivation of ACV

- Equation determining the weights

$$\nabla \mathcal{H}(\boldsymbol{w}) = 0 \Rightarrow \hat{\boldsymbol{w}}$$

$$\nabla \mathcal{H}^{\backslash \mu}(\boldsymbol{w}) = 0 \Rightarrow \hat{\boldsymbol{w}}^{\backslash \mu}$$

- Perturbation assuming the smallness of $\boldsymbol{d}_\mu = \hat{\boldsymbol{w}} - \hat{\boldsymbol{w}}^{\backslash \mu}$

$$\begin{aligned}
0 = \nabla \mathcal{H}^{\backslash \mu}(\hat{\boldsymbol{w}}^{\backslash \mu}) &= \nabla \mathcal{H}^{\backslash \mu}(\hat{\boldsymbol{w}} - \boldsymbol{d}_\mu) \approx \nabla \mathcal{H}^{\backslash \mu}(\hat{\boldsymbol{w}}) - \partial^2 \mathcal{H}^{\backslash \mu}(\hat{\boldsymbol{w}}) \boldsymbol{d}_\mu \\
&= \nabla (\mathcal{H}(\hat{\boldsymbol{w}}) - q_\mu(\hat{\boldsymbol{w}})) - \partial^2 \mathcal{H}^{\backslash \mu}(\hat{\boldsymbol{w}}) \boldsymbol{d}_\mu \\
&= -\nabla q_\mu(\hat{\boldsymbol{w}}) - G^{\backslash \mu}(\hat{\boldsymbol{w}}) \boldsymbol{d}_\mu, \\
\Rightarrow \quad \boldsymbol{d}_\mu &\approx -\left(G^{\backslash \mu}(\hat{\boldsymbol{w}})\right)^{-1} \nabla q_\mu(\hat{\boldsymbol{w}}).
\end{aligned}$$

$$G^{\backslash \mu}(\hat{\boldsymbol{w}}) = G(\hat{\boldsymbol{w}}) - \partial^2 q_\mu(\hat{\boldsymbol{w}}),$$

$$\Rightarrow \boxed{\hat{\boldsymbol{w}}^{\backslash \mu} = \hat{\boldsymbol{w}} - \boldsymbol{d}_\mu \approx \hat{\boldsymbol{w}} + \left(G(\hat{\boldsymbol{w}}) - \partial^2 q_\mu(\hat{\boldsymbol{w}})\right)^{-1} \nabla q_\mu(\hat{\boldsymbol{w}})}$$

Derivation of ACV

- Multiplying $\boldsymbol{x}_\mu^\top$ and simplifying factors by Sherman-Morrison formula

$$\hat{\boldsymbol{w}}^{\setminus\mu} = \hat{\boldsymbol{w}} - \boldsymbol{d}_\mu \approx \hat{\boldsymbol{w}} + \left(G(\hat{\boldsymbol{w}}) - \partial^2 q_\mu(\hat{\boldsymbol{w}})\right)^{-1} \nabla q_\mu(\hat{\boldsymbol{w}})$$

$$\Rightarrow \quad \hat{\boldsymbol{u}}_\mu^{\setminus\mu} \approx \hat{\boldsymbol{u}}_\mu + C_\mu \left(I - C_\mu F_\mu\right)^{-1} \boldsymbol{b}_\mu$$

- To apply the derivation so far to the case with sparse regularizations, the computation of gradient and Hessian is done only on the active set $\hat{A} = \{(ai) | \hat{w}_{ai} \neq 0\}$
  - We also need to assume that the active set is common between full and LOO systems
    - This is ``approximately'' true
      - Negligible contributions by variables come and go support

Warning

- The factor $C_\mu = X^\mu_{*\hat{A}} \left(G_{\hat{A}\hat{A}}\right)^{-1} \left(X^\mu_{*\hat{A}}\right)^\top$ is *not well defined*
  - Because the Hessian G has zero eigenvalues
    - The model is singular because the MLR is invariant against

$$\boldsymbol{w}_a \to \boldsymbol{w}_a + \boldsymbol{v} \ (\forall a)$$

    - This is gauge degree of freedom (gdf) in physics
  - Some implementation erases this gdf by fixing certain class weights at zeros (Krishnapuram et al., 2005; Schmidt, 2010)
    - This is nice for our approximation
  - But not all implementations do this
    - E.g. *Glmnet* (Friedman et al., 2010) doesn't

→Another way should be tailored to avoid the problem

Zero Mode Removal by Hand

- The factor $C_\mu = X^\mu_{*\hat{A}} \left(G_{\hat{A}\hat{A}}\right)^{-1} \left(X^\mu_{*\hat{A}}\right)^\top$ is *not well defined*
  - Because the Hessian G has zero eigenvalues
    - Remove the zero modes by hand

$$G_{\hat{A}\hat{A}} \overset{\text{EVD}}{=} \sum_i d_i \boldsymbol{v}_i \boldsymbol{v}_i^\top = \sum_{i \in S^+} d_i \boldsymbol{v}_i \boldsymbol{v}_i^\top ,$$

$$\Rightarrow \quad \overline{G}^{-1}_{\hat{A}\hat{A}} \equiv \sum_{i \in S^+} d_i^{-1} \boldsymbol{v}_i \boldsymbol{v}_i^\top .$$

$$\Rightarrow \quad C_\mu = X^\mu_{*\hat{A}} \overline{G}^{-1}_{\hat{A}\hat{A}} \left(X^\mu_{*\hat{A}}\right)^\top .$$

This seems to be ad-hoc but actually works!

Algorithm and Cost (ACV)

Algorithm 1 Approximate CV of the MLR

procedure ACV($\hat{\boldsymbol{W}}(\lambda), D^M$)
  Compute the active set $\hat{A}$ from $\hat{\boldsymbol{W}}$
  Compute $\{\hat{\boldsymbol{u}}_\mu, X^\mu, \boldsymbol{b}^\mu, F^\mu\}_\mu$ by eqs. (1,7), (12) and (13)
  $G_{\hat{A}\hat{A}} \leftarrow \sum_{\mu=1}^{M} (X^\mu)^\top F^\mu X^\mu$ ▷ $O(ML|\hat{A}|^2 + ML^2|\hat{A}|)$
  Compute $\overline{G}^{-1}_{\hat{A}\hat{A}}$ by eq. (25) ▷ $O(|\hat{A}|^3)$
  for $\mu = 1, \cdots, M$ **do** ▷ $O(ML|\hat{A}|^2 + ML^2|\hat{A}| + ML^3)$
    $C_\mu \leftarrow X^\mu_{*\hat{A}} \overline{G}^{-1}_{\hat{A}\hat{A}} \left(X^\mu_{*\hat{A}}\right)^\top$
    $\hat{\boldsymbol{u}}_\mu^{\backslash\mu} \leftarrow \hat{\boldsymbol{u}}_\mu + C_\mu \left(I_L - F^\mu C_\mu\right)^{-1} \boldsymbol{b}^\mu$
  end for
  Compute ϵ_{LOO} from $\{\boldsymbol{u}_\mu^{\backslash\mu}\}_\mu$ by eq. (6)
  return ϵ_{LOO}
end procedure

- Comp. cost$= O(ML|\hat{A}|^2 + |\hat{A}|^3) \approx O(MN^2 + N^3)$
  - Too heavy if *N,M* are large

Self-Averaging Approximation (SAACV)

- Neglecting ``correlations'' bet. different feature components

$$\left(G^{\setminus\mu}\right)^{-1}_{(ai)(bj)} \approx \begin{cases} (\chi_i)_{ab}\,\delta_{ij}, & ((ai),(bj)\in\hat{A}) \\ 0, & (\text{otherwise}) \end{cases},$$

$\chi_i \in \mathbb{R}^{L\times L}$: Inter-class susceptibility (rescaled variance)

- Self-consistent equation to determine χ

$$\begin{cases} (\chi_i)_{\hat{A}_i\hat{A}_i} = \dfrac{1}{\sigma_x^2}\left(\displaystyle\sum_{\nu=1}^{M}\left(\left(I_L + F^\nu C_{\mathrm{SA}}\right)^{-1} F^\nu\right)_{\hat{A}_i\hat{A}_i}\right)^{-1} \\ C_{\mathrm{SA}} = \sigma_x^2 \displaystyle\sum_{i=1}^{N}\chi_i \end{cases}$$

$$\hat{A}_i = \{a|\hat{w}_{ai}\neq 0\} \quad \sigma_x^2 = \frac{1}{NM}\sum_{\mu,i} x_{\mu i}^2$$

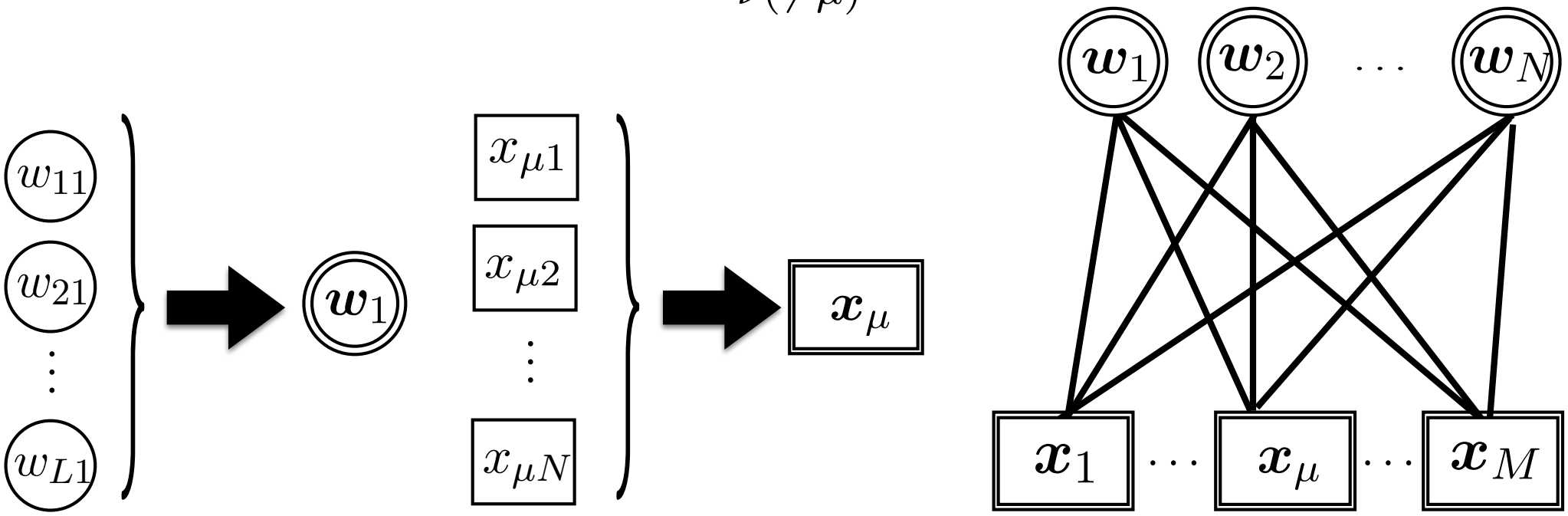
- Computational cost: $O(NL^3 + ML^3)$ *Linear in N and M!*

Derivation of SAACV

- Cavity method/Belief propagation for multicomponent model

$$\tilde{M}_{\mu\to i}(\boldsymbol{w}_i) = \int \prod_{j(\neq i)} d\boldsymbol{w}_j \; \phi^{\beta}(\boldsymbol{u}_\mu) \prod_{j(\neq i)} M_{j\to\mu}(\boldsymbol{w}_j),$$

$$M_{i\to\mu}(\boldsymbol{w}_i) = e^{-\beta\lambda||\boldsymbol{w}_i||_1} \prod_{\nu(\neq\mu)} \tilde{M}_{\nu\to i}(\boldsymbol{w}_i),$$



Derivation of SAACV

- Cavity method/Belief propagation for multicomponent model

$$\tilde{M}_{\mu\to i}(\boldsymbol{w}_i) = \int \prod_{j(\neq i)} d\boldsymbol{w}_j \; \phi^{\beta}(\boldsymbol{u}_\mu) \prod_{j(\neq i)} M_{j\to\mu}(\boldsymbol{w}_j),$$

$$u_{\mu a} = \sum_i x_{\mu i} w_{ai} \approx x_{\mu i} w_{ai} + \sum_{j(\neq i)} x_{\mu j} \left\langle w_{aj} \right\rangle^{\backslash \mu} + t_a, \;\; \boldsymbol{t} \sim \mathcal{N}\left(0, \mathrm{Cov}\left(\sum_{j(\neq i)} x_{\mu j} \boldsymbol{w}_j\right)\right)$$

$$\chi^{\backslash\mu}_{(ai)(bj)} \equiv \beta\left(\left\langle w_{ai} w_{bj} \right\rangle^{\backslash\mu} - \left\langle w_{ai} \right\rangle^{\backslash\mu} \left\langle w_{bj} \right\rangle^{\backslash\mu}\right) \approx \delta_{ij} \boxed{\beta\left(\left\langle w_{ai} w_{bi} \right\rangle^{\backslash\mu} - \left\langle w_{ai} \right\rangle^{\backslash\mu} \left\langle w_{bi} \right\rangle^{\backslash\mu}\right)}$$

$$\left(C^{\backslash\mu}_{\mu}\right)_{ab} \equiv \sum_{i,j} x_{\mu i} x_{\mu j} \chi^{\backslash\mu}_{(ai)(bj)} \approx \sum_i x^2_{\mu i} \boxed{\left(\chi^{\backslash\mu}_i\right)_{ab}}$$

$$\tilde{M}_{\mu\to i}(\boldsymbol{w}_i) \approx \int d\boldsymbol{t} \; e^{\beta\left(-\frac{1}{2}\boldsymbol{t}^\top (C^{\backslash\mu}_{\mu})^{-1}\boldsymbol{t} - q_\mu(\boldsymbol{w}_i, \boldsymbol{t})\right)} \equiv \int d\boldsymbol{t} \; e^{\beta f^{\mu}(\boldsymbol{w}_i, \boldsymbol{t})}$$

- The remaining derivation is the same as usual BP

Algorithm and Cost (SAACV)

Algorithm 2 Self-averaging approximate CV of the MLR

procedure SAACV($\hat{\boldsymbol{W}}(\lambda), D^M$)
  Compute the active sets $\{\hat{A}_i\}_{i=1}^N$ from $\hat{\boldsymbol{W}}$
  Compute $\{\boldsymbol{u}_\mu, X^\mu, \boldsymbol{b}_\mu, F^\mu\}_\mu$ by eqs. (1,7), (12) and (13)
  $t \leftarrow 0$ ▷ Start initialization
  for $i = 1, \cdots, N$ **do**
    $\left(\chi_i^{\setminus\mu}\right)^{(t)} \leftarrow 0,$
    $\left(\chi_i^{\setminus\mu}\right)^{(t)}_{\hat{A}_i\hat{A}_i} \leftarrow \sigma_x^{-2},$
  end for
  $\Delta \leftarrow 100$ ▷ End initialization
  while $\Delta > \theta$ **do** ▷ Compute C_{SA} by recursion
    $C_{\mathrm{SA}}^{(t+1)} \leftarrow \sigma_x^2 \sum_{i=1}^N \left(\chi_i^{\setminus\mu}\right)^{(t)}$
    $R \leftarrow \sum_{\mu=1}^M \left(I_L + F^\mu C_{\mathrm{SA}}^{(t+1)}\right)^{-1} F^\mu$ ▷ $O(ML^3)$
    $\Delta \leftarrow 0$
    for $i = 1, \cdots, N$ **do** ▷ $O(NL^3)$
      Compute $\overline{R}^{-1}_{\hat{A}_i\hat{A}_i}$ by eq. (41) from R
      $\left(\chi_i^{\setminus\mu}\right)^{(t+1)}_{\hat{A}_i\hat{A}_i} \leftarrow \sigma_x^{-2}\overline{R}^{-1}_{\hat{A}_i\hat{A}_i}$
      $\Delta \leftarrow \Delta + \left\| \left(\chi_i^{\setminus\mu}\right)^{(t+1)}_{\hat{A}_i\hat{A}_i} - \left(\chi_i^{\setminus\mu}\right)^{(t)}_{\hat{A}_i\hat{A}_i} \right\|_{\mathrm{F}}$
    end for
    $\Delta \leftarrow \Delta / N$
    $t \leftarrow t + 1$
  end while
  for $\mu = 1, \cdots, M$ **do**
    $\boldsymbol{u}_\mu^{\setminus\mu} \leftarrow \boldsymbol{u}_\mu + C_{\mathrm{SA}}^{(t)} \boldsymbol{b}^\mu$
  end for
  Compute ϵ_{LOO} from $\{\boldsymbol{u}_\mu^{\setminus\mu}\}_\mu$ by eq. (6)
  return ϵ_{LOO}
end procedure

Simulated Data

- True feature vector is sparse

$$\boldsymbol{w}_{0a} \sim \prod_{i=1}^{N} \{(1-\rho_0)\delta(w_{0ai}) + \rho_0 \mathcal{N}(0, 1/\rho_0)\}$$

- Class label y_μ is drawn uniformly and randomly

$$y_\mu \sim (1/L) \sum_{a=1}^{L} \delta_{a y_\mu}$$

- Observed Feature=True Feature+Noise

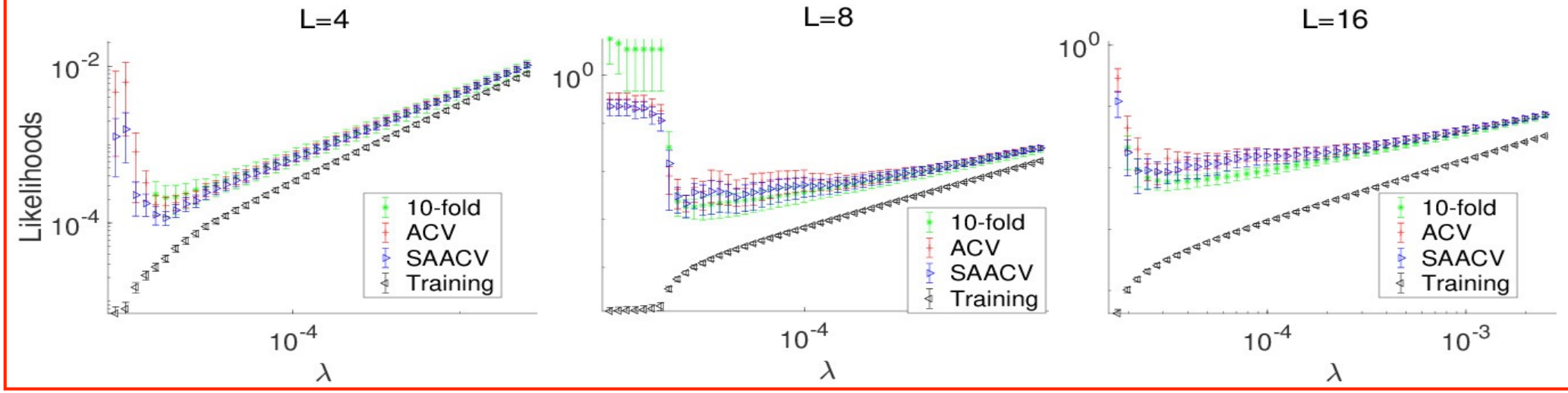
$$\boldsymbol{x}_\mu = \frac{\boldsymbol{w}_{0 y_\mu}}{\sqrt{N}} + \boldsymbol{\xi}, \quad \xi_i \sim \mathcal{N}(0, \sigma_N^2)$$

- The overall parameters= $\{N, L, \alpha, \rho_0, \sigma_\xi^2\}$, $\left(\alpha = \frac{M}{N}\right)$

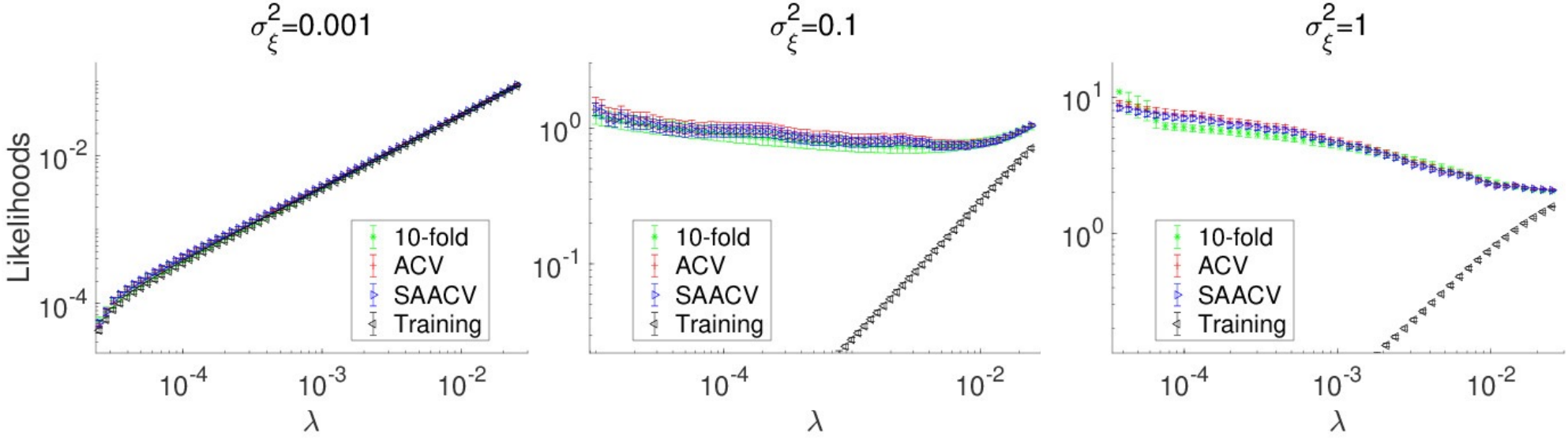
Fix $\alpha = 2, \rho_0 = 0.5$ for simplicity below

Dependence on Class Number and Noise Strength

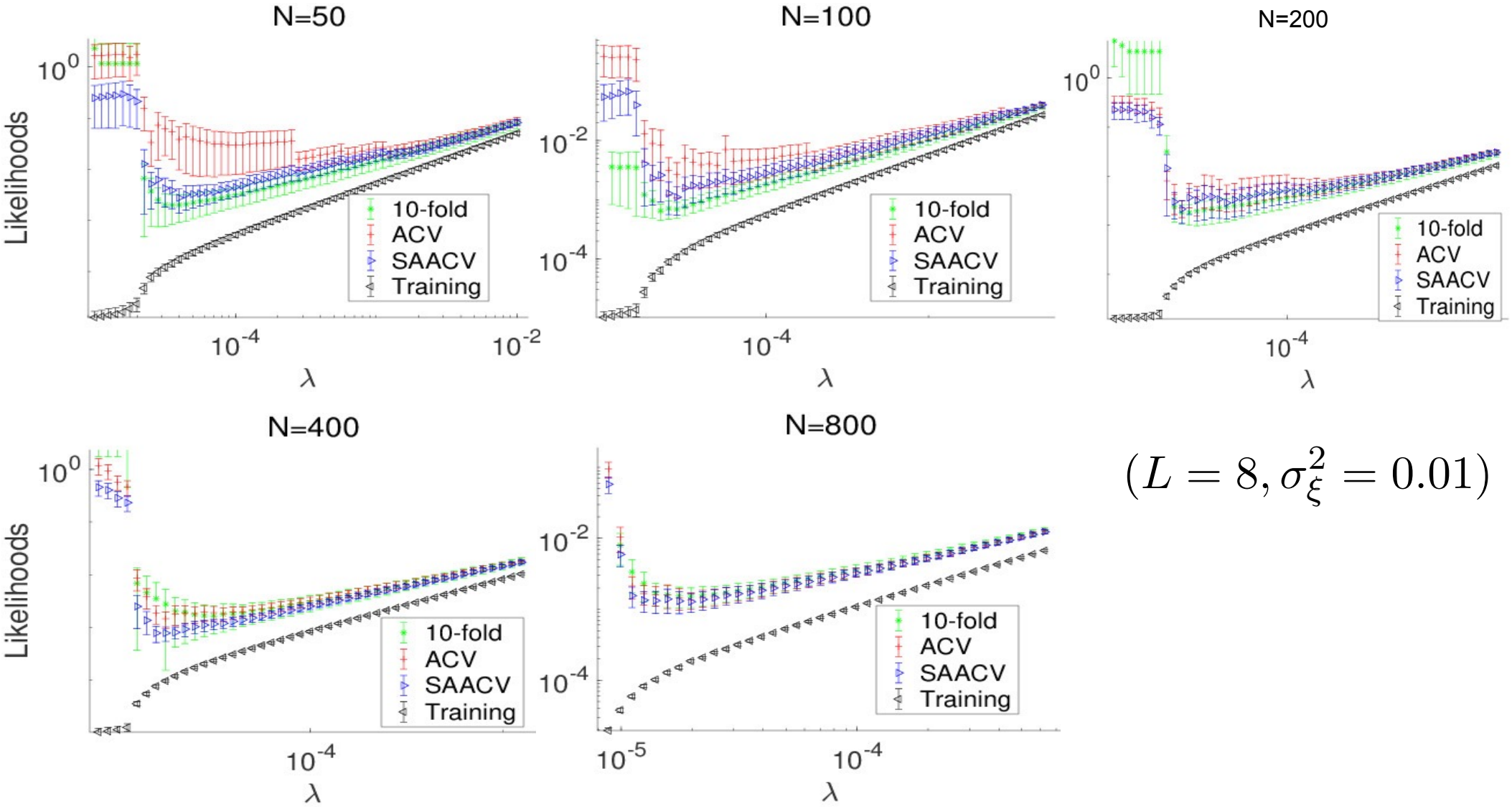
$(N = 200, \sigma_\xi^2 = 0.01)$



$(N = 200, L = 8)$



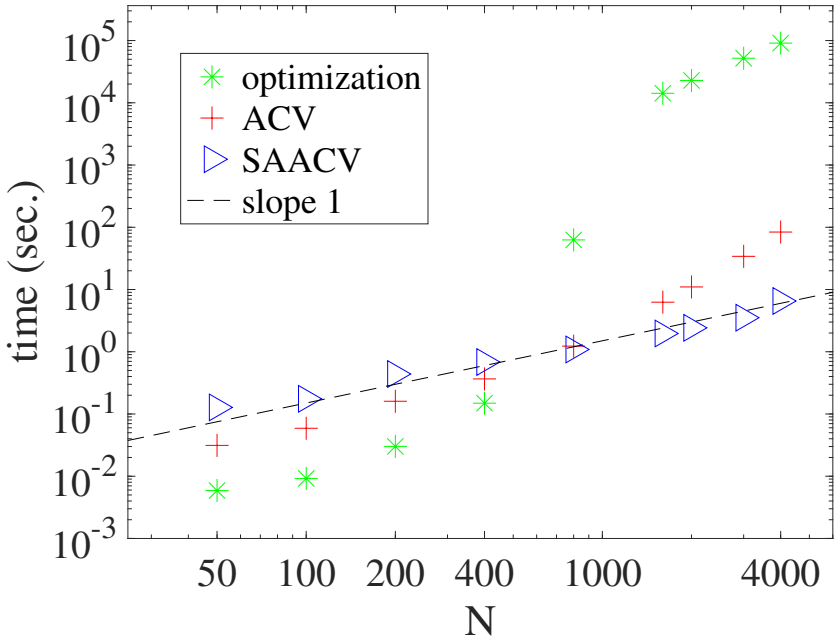
System-Size Dependence



$(L = 8, \sigma_\xi^2 = 0.01)$

Accuracy seems to be good for $N \geq 200$ even at small λs.

Computational Time



- Optimization is done by Glmnet
  - A specialized version of coordinate descent method
    - The comp. cost is supposed to be O(N^2)

- Small (N<400): Opt < ACV < SAACV → Literal CV
- Middle (400≦N<800): ACV < SAACV≪ Opt → ACV
- Large (800≦N): SAACV ≪ ACV ≪ Opt → SAACV

ISOLET DATA

ISOLET Data Set

Download: Data Folder, Data Set Description

Abstract: Goal: Predict which letter-name was spoken--a simple classification task.

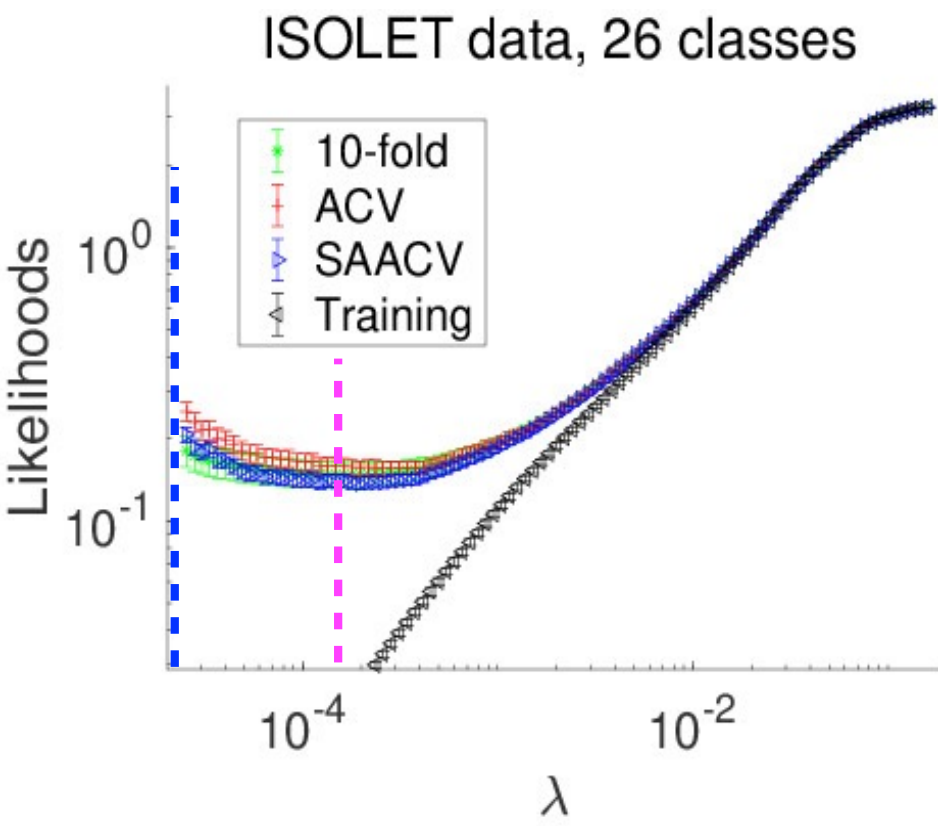
Data Set Characteristics:	Multivariate	**Number of Instances:**	7797	**Area:**	Computer
Attribute Characteristics:	Real	**Number of Attributes:**	617	**Date Donated**	1994-09-12
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	82016

Data Set Information:

This data set was generated as follows. 150 subjects spoke the name of each letter of the alphabet twice. Hence, we have 52 training examples from each speaker. The speakers are grouped into sets of 30 speakers each, and are referred to as isolet1, isolet2, isolet3, isolet4, and isolet5. The data appears in

- Classification task of N=617, M=6238, and L=26

ISOLET DATA



Actual computational time:

10-fold	ACV	SAACV
7825 s	5173 s	689 s

Accuracy rate at the pred. llk. minimum:

10-fold	ACV	SAACV
0.86	0.86	0.86

Accuracy rate at the minimum λ:

10-fold	ACV	SAACV
0.83	0.78	0.81

- ISOLET: ℓ_1 regularization is meaningful
  - N,M,L are large→ACV is not so good, SAACV is great :)

Summary

TO, YK: arXiv: 1711.05420

- An approximate formula of CV for the MLR with the ℓ_1 regularization is developed.
  - Perturbation using the largeness of model&dataset
  - Two versions: ACV, SAACV
    - ACV comp. cost: $O(ML|\hat{A}|^2 + |\hat{A}|^3) \approx O(MLN^2 + N^3)$
    - SAACV comp. cost: $O(ML^3 + NL^3)$ (linear in *M&N*)
      - Cavity method for multicomponent model is used
        - Block-wise correlations are taken into account
- Numerical experiments demonstrate when these approximations are good and advantageous
- Matlab, python implementations are available from
  https://github.com/T-Obuchi/AcceleratedCVonMLR_matlab
  https://github.com/T-Obuchi/AcceleratedCVonMLR_python

  Elastic net (ℓ_1+ℓ_2) regularization is also covered in the implementation