Iterative Approximate Cross Validation in High Dimensions

Sina Alsharif

School of Computer Science and Engineering, University of New South Wales

Thesis B (Term 3, 2023)

Section 1

Approximate CV Methods

Approximate CV Methods

To reduce the computational cost of CV (specifically LOOCV), we turn to Approximate Cross Validation (ACV), which attempts to approximate (rather than solve) individual CV experiments.

LOOCV

The definition of leave-one-out (LOO) regularised empirical risk is,

$$R_{\text{reg}}(\theta; D_{-j}) = \sum_{i=1,\, i \neq j}^{n} \ell(\theta; D_i) + \lambda \pi(\theta)$$

where we leave out a point with index j for this experiment.

Quick Recap

There are three main methods for ACV we will discuss.

- Newton Step ("NS")
- Infinitesimal Jackknife ("IJ")
- Iterative Approximate Cross Validation ("IACV")

both NS and IJ are existing methods, and IACV is a new proposed method which we aim to adapt and extend.

Quick Recap

There are three main methods for ACV we will discuss.

- Newton Step ("NS")
- Infinitesimal Jackknife ("IJ")
- Iterative Approximate Cross Validation ("IACV")

both NS and IJ are existing methods, and IACV is a new proposed method which we aim to adapt and extend.

Newton Step

For discussion, the standard notation we'll use for the NS method is,

$$\tilde{\theta}_{\mathsf{NS}}^{-i} = \hat{\theta} + \left(H(\hat{\theta}; D) - \nabla_{\theta}^{2} \ell(\hat{\theta}; D_i) \right)^{-1} \nabla_{\theta} R_{\mathsf{reg}}(\hat{\theta}; D_i)$$

Infinitesimal Jackknife

The final form derived for the IJ estimator is

$$\tilde{\theta}_{\mathsf{IJ}}^{-i} = \hat{\theta} + (H(\hat{\theta}; D))^{-1} \nabla_\theta R_{\mathsf{reg}}(\hat{\theta}; D_i)$$

where we again make the same assumptions as in NS (loss and regularisation are continuously twice-differentiable, H is invertible).

Infinitesimal Jackknife

The final form derived for the IJ estimator is

$$\tilde{\theta}_{\mathsf{IJ}}^{-i} = \hat{\theta} + (H(\hat{\theta}; D))^{-1} \nabla_{\theta} R_{\mathsf{reg}}(\hat{\theta}; D_i)$$

where we again make the same assumptions as in NS (loss and regularisation are continuously twice-differentiable, H is invertible). This method has a computational advantage over NS, as we only need to calculate and invert $H(\hat{\theta}; D)$ once, rather than n times.

Iterative Approximate Cross Validation

Iterative Approximate Cross Validation (IACV) relaxes these assumptions and gives a more general form.

We again solve the main learning task through an iterative method, where the updates are (for GD and SGD)

$$\hat{\theta}^{(k)} = \hat{\theta}^{(k-1)} - \alpha_k \nabla_\theta R_{\text{reg}}(\hat{\theta}^{(k-1)}; D_{S_k})$$

where $S_k \subseteq [n]$ is a subset of indices and α_k is a learning rate taken for an iteration k. For classic GD, $S_k \equiv [n]$ and can be variable for SGD.

Iterative Approximate Cross Validation

Iterative Approximate Cross Validation (IACV) relaxes these assumptions and gives a more general form.

We again solve the main learning task through an iterative method, where the updates are (for GD and SGD)

$$\hat{\theta}^{(k)} = \hat{\theta}^{(k-1)} - \alpha_k \nabla_\theta R_{\text{reg}}(\hat{\theta}^{(k-1)}; D_{S_k})$$

where $S_k \subseteq [n]$ is a subset of indices and α_k is a learning rate taken for an iteration k. For classic GD, $S_k \equiv [n]$ and can be variable for SGD.

The explicit optimisation step LOOCV iterate excluding a point i is defined as,

$$\hat{\theta}_{-i}^{(k)} = \hat{\theta}_{-i}^{(k-1)} - \alpha_k \nabla_\theta R_{\text{reg}}(\hat{\theta}_{-i}^{(k-1)}; D_{S_k \setminus i})$$

this step is what we aim to approximate.

We use a second-order Taylor expansion to reduce the computation burden of calculating the Jacobian for each point:

$$\nabla_\theta R_{\text{reg}}(\hat{\theta}_{-i}^{(k-1)}; D_{S_k \setminus i}) \approx \nabla_\theta R_{\text{reg}}(\hat{\theta}^{(k-1)}; D_{S_k \setminus i}) + \nabla_\theta^2 R_{\text{reg}}(\hat{\theta}^{(k-1)}; D_{S_k \setminus i}) \left(\tilde{\theta}_{-i}^{(k-1)} - \hat{\theta}^{(k-1)}\right)$$

is the estimate for the Jacobian.

We use a second-order Taylor expansion to reduce the computation burden of calculating the Jacobian for each point:

$$\nabla_\theta R_{\text{reg}}(\hat{\theta}_{-i}^{(k-1)}; D_{S_k \setminus i}) \approx \nabla_\theta R_{\text{reg}}(\hat{\theta}^{(k-1)}; D_{S_k \setminus i}) + \nabla_\theta^2 R_{\text{reg}}(\hat{\theta}^{(k-1)}; D_{S_k \setminus i}) \left(\tilde{\theta}_{-i}^{(k-1)} - \hat{\theta}^{(k-1)}\right)$$

is the estimate for the Jacobian.

Therefore, the IACV updates for GD and SGD become,

$$\tilde{\theta}_{-i}^{(k)} = \tilde{\theta}_{-i}^{(k-1)} - \alpha_k \left(\nabla_\theta R_{\text{reg}}(\hat{\theta}^{(k-1)}; D_{S_k \setminus i}) + \nabla_\theta^2 R_{\text{reg}}(\hat{\theta}^{(k-1)}; D_{S_k \setminus i}) \left(\tilde{\theta}_{-i}^{(k-1)} - \hat{\theta}^{(k-1)}\right)\right)$$

The main difference (from NS and IJ) is that we can define an ACV update rule for proximal gradient descent.

The main difference (from NS and IJ) is that we can define an ACV update rule for proximal gradient descent. If we define a general update rule for LOOCV proximal gradient descent as,

$$\hat{\theta}_{-i}^{(k)} = \arg\min_{z} \left\{ \frac{1}{2\alpha_k} \|z - \theta'_{-i}\|_2^2 + \lambda\pi(z) \right\}$$
$$\text{where } \theta'_{-i} = \hat{\theta}_{-i}^{(k-1)} - \alpha_k \nabla_\theta \ell(\hat{\theta}_{-i}^{(k-1)}; D_{S_k \setminus i})$$

using similar logic as in GD/SGD on the differentiable part of the regularised risk, we get IACV updates of,

The main difference (from NS and IJ) is that we can define an ACV update rule for proximal gradient descent. If we define a general update rule for LOOCV proximal gradient descent as,

$$\hat{\theta}_{-i}^{(k)} = \arg\min_{z} \left\{ \frac{1}{2\alpha_k} \|z - \theta'_{-i}\|_2^2 + \lambda\pi(z) \right\}$$
$$\text{where } \theta'_{-i} = \hat{\theta}_{-i}^{(k-1)} - \alpha_k \nabla_\theta \ell(\hat{\theta}_{-i}^{(k-1)}; D_{S_k \setminus i})$$

using similar logic as in GD/SGD on the differentiable part of the regularised risk, we get IACV updates of,

$$\tilde{\theta}_{-i}^{(k)} = \arg\min_{z} \left\{ \frac{1}{2\alpha_k} \|z - \theta'_{-i}\|_2^2 + \lambda\pi(z) \right\}$$
$$\text{where } \theta'_{-i} = \tilde{\theta}_{-i}^{(k-1)} - \alpha_k \left(\nabla_\theta \ell(\hat{\theta}^{(k-1)}; D_{S_k \setminus i}) + \nabla_\theta^2 \ell(\hat{\theta}^{(k-1)}; D_{S_k \setminus i}) \left(\tilde{\theta}_{-i}^{(k-1)} - \hat{\theta}^{(k-1)} \right) \right)$$

The time complexities for the operations are as follows,

	IACV	Exact LOOCV
GD	$n(A_p + B_p) + np^2$	$n^2 A_p + np$
SGD	$K(A_p + B_p) + np^2$	$nKA_p + np$
ProxGD	$n(A_p + B_p + D_p) + np^2$	$n^2 A_p + nD_p + np$

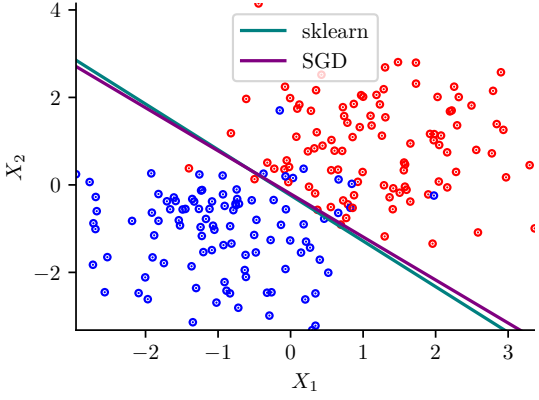
with breakdown as p increases.

Section 2

Experimental Work

Experimental Work

We experimented with applying IACV to SVM solved through SGD. I recreated the algorithm shown in Shai Shalev-Schwartz's text which solves soft-margin SVM through SGD.



Applying IACV

Trying to apply IACV to the SVM algorithm led me to looking a *smooth* hinge loss [Luo21], which allows for the calculation of the Hessian required for IACV. The simplest form representing this is as follows;

Applying IACV

Trying to apply IACV to the SVM algorithm led me to looking a *smooth* hinge loss [Luo21], which allows for the calculation of the Hessian required for IACV. The simplest form representing this is as follows;

Define

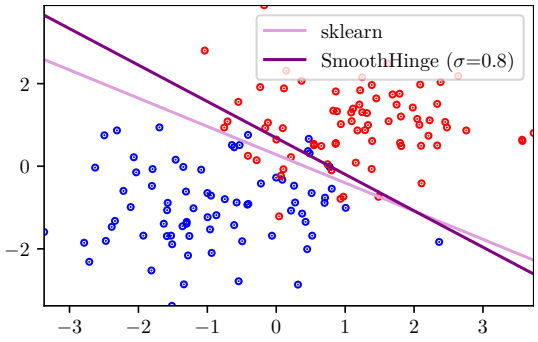
$$\Phi_m(v) = (1 + v/\sqrt{(1+v^2)})/2$$
$$\phi_m(v) = 1/(2 \cdot \sqrt{(1+v^2)}).$$

For a final form of

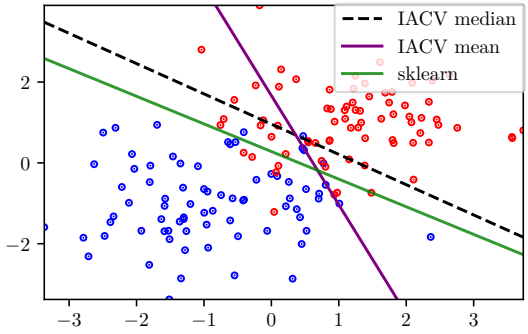
$$\psi_m(v) = \Phi_m(v/\sigma) \cdot v + \phi_m(v/\sigma) \cdot \sigma,$$

where σ controls how close the final result is to the original hinge loss (i.e as $\sigma \to 0$ we approach the classic hinge loss).

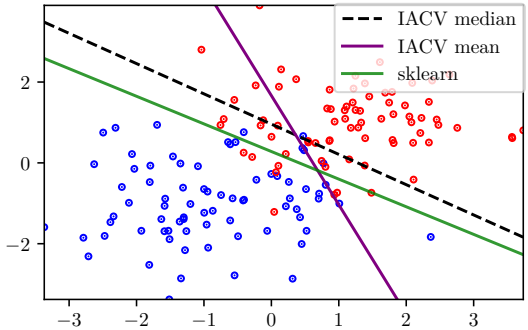
My implementation gives reasonable results for $\sigma = 0.8$ (this choice of σ will be explained later).



After applying IACV, we retrieve the following statistics from the LOOCV iterates.

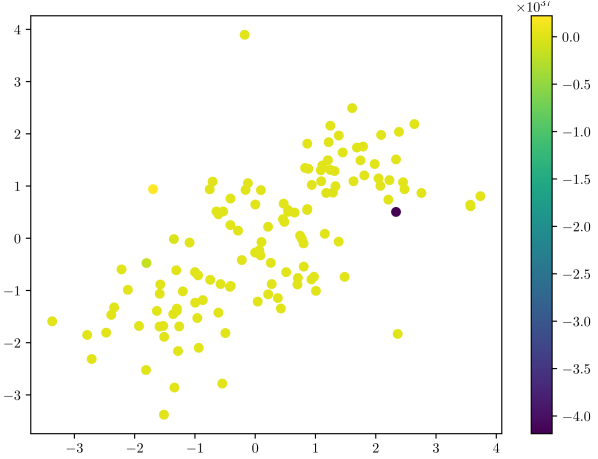


After applying IACV, we retrieve the following statistics from the LOOCV iterates.



Observe that the mean gives no meaningful information, and indicates a numerical issue.

After analysing the sum of the weights when leaving a point out, we see anomalies (shown in purple) which impact the mean, confidence intervals and overall utility of the cross-validation. This phenomenon becomes more apparent (more points with numerical issues) as $\sigma \to 0$.



Section 3

Back to High Dimensions

Back to High Dimensions

Approximate cross validation tends to break down in high dimensions. For IJ and NS, the main problems are

- Time complexity breakdown (especially for Hessian inversion)
- A breakdown of accuracy

Where a similar theme follows somewhat for IACV.

Existing Solutions for High Dimensional Problems

The current solutions for IJ and NS are to use the support of the ℓ-1 solution at each iteration to reduce to the computation cost and increase the accuracy of the method. For the estimated support $\hat{S}$, we have

$$\left[\tilde{\theta}_{\text{NS}}^{-i}\right]_j = \begin{cases} 0 & \text{when } \hat{\theta}_j = 0 \\ \hat{\theta}_j + \left[\left(H_{\hat{S}}(\hat{\theta}_{\hat{S}}; D) - \nabla^2_{\theta}\ell_{\hat{S}}(\hat{\theta}_{\hat{S}}; D_{-i})\right)^{-1} \nabla_{\theta}\ell_{\hat{S}}(\hat{\theta}_{\hat{S}}; D_i)\right]_j & \text{otherwise} \end{cases}$$

where we only evaluate the terms in the support.

Similarly, the "sparse ACV" updates for IJ becomes,

$$\left[\tilde{\theta}_{\text{IJ}}^{-i}\right]_j = \begin{cases} 0 & \text{when } \hat{\theta}_j = 0 \\ \hat{\theta}_j + \left[\left(H_{\hat{S}}(\hat{\theta}_{\hat{S}}; D)\right)^{-1} \nabla_{\theta}\ell_{\hat{S}}(\hat{\theta}_{\hat{S}}; D_i)\right]_j & \text{otherwise.} \end{cases}$$

The paper which proposes this solution is [SB20]. I've had a go at recreating the results in their paper, with mild success. Running LASSO (for linear regression) for an experiment with $p = 150$ and $n = 50$ as,

Method	Err_{App}	Err_{LOO}
Baseline ($\hat{\theta}$)	4.25	109.56
IJ (sparse)	4.16	0.35
NS (sparse)	4.20	0.62
IACV	12.3	54.8

Where $\text{Err}_{\text{App}} = \frac{1}{n}\sum_{i=1}^{n} \|\hat{\theta}_{-i} - \tilde{\theta} - i\|_2^2$ and $\text{Err}_{\text{LOO}} = \sum_{i=1}^{n} |\ell(\tilde{\theta}_{-i}) - \ell(\hat{\theta}_{-i})|/\ell(\hat{\theta}_{-i})$.

The original paper has errors in the same orders of magnitude.

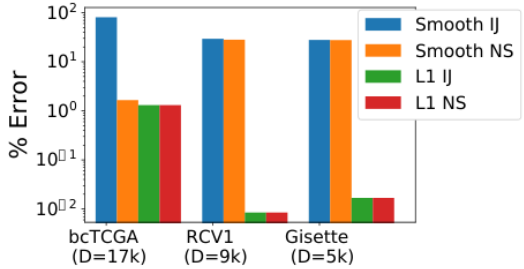


Figure 1: Error rates for smoothed and "sparse" (L1) ACV methods on real data [SB20].

The next step is to fully recreate the experiments done in the original paper, however the specific datasets they used are no longer available.

The original paper has errors in the same orders of magnitude.

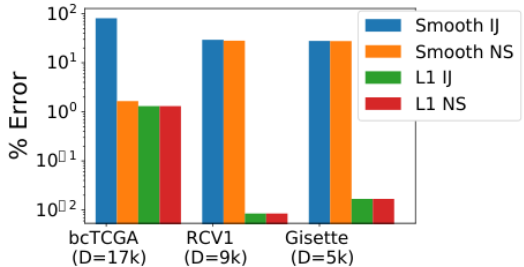


Figure 1: Error rates for smoothed and "sparse" (L1) ACV methods on real data [SB20].

The next step is to fully recreate the experiments done in the original paper, however the specific datasets they used are no longer available. Also, the accuracy of these methods depends highly on the condition $\lambda \geq c\sqrt{\frac{\log(p)}{n}}$ for some c we pick ($c = n/p$ seems to work fine).

They use coordinate descent!

They use coordinate descent!

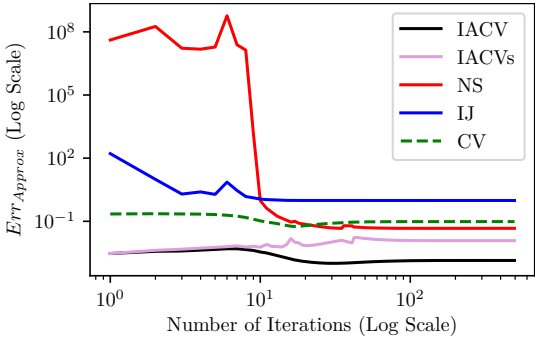
This means we cannot directly apply IACV to the same problem, I have tried using the proximal descent (ISTA) implementation to update IACV iterates alongside the main coordinate descent to no success.

Applying Sparse ACV to IACV

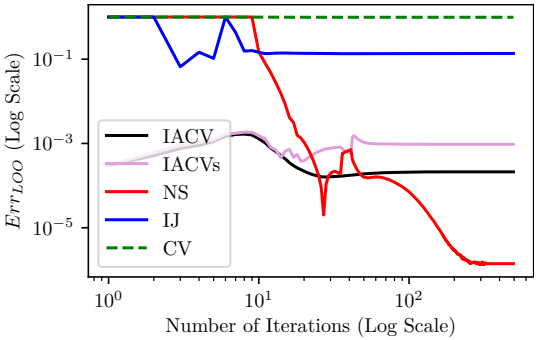
I have however, applied the 'sparse' ACV method presented previously to IACV and the other ACV methods to a logistic lasso task, with $p = 150$ and $n = 50$, solved through ISTA.

Applying Sparse ACV to IACV

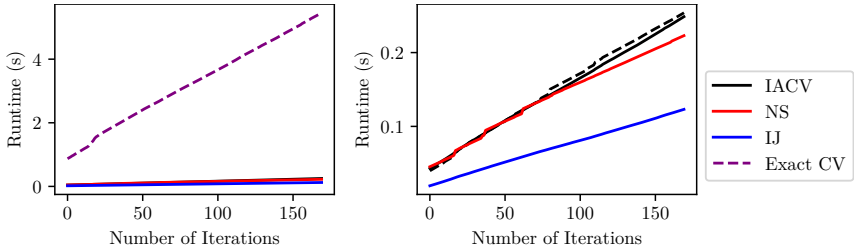
I have however, applied the 'sparse' ACV method presented previously to IACV and the other ACV methods to a logistic lasso task, with $p = 150$ and $n = 50$, solved through ISTA.



The percentage LOO error is as follows,



The runtime looks something like this.



Section 4

Future Plans

Future Plans

My plans are to stick with improving IACV for high dimensions, since that seems to have the most ground for improvement and exploration. Things I would like to get done in the upcoming break and Thesis C are,

- General code clean up.
- Experiment with different problem settings for IACV.
- Recreate the Gisette experiment in the Broderick paper (only recently found usable data for this).
- Modify the existing experiment code for a speed-up.
- Look into the theory - why is IACV better at approximating coefficients rather than the CV loss?
- Is there an IACV-like method for coordinate descent?