Iterative Approximate Cross Validation in High Dimensions

Sina Alsharif

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

Thesis C (Term 1, 2024)

Section 1

Background

Background

To standardise the notation used throughout this seminar, we define the Empirical Risk Minimisation (ERM) framework used to solve a supervised learning problem.

Background

To standardise the notation used throughout this seminar, we define the Empirical Risk Minimisation (ERM) framework used to solve a supervised learning problem.

General Problem Setting

- Input space $\mathcal{X}$ and an output space $\mathcal{Y}$.
- Data is "generated" by a *true* distribution $P(\mathcal{X}, \mathcal{Y})$.
- Find a mapping $h : \mathcal{X} \to \mathcal{Y}$ (called a hypothesis).
- All possible combinations of input and output space are $\mathcal{D} = \{(X, Y) \in (\mathcal{X}, \mathcal{Y})\}$.

Risk

To measure the error (or loss) we make on a data point, define a function $\ell(h; D_i)$,

$$\ell(h; D) = \sum_{i=1}^{n} \ell(h; D_i) \text{ where } D_i = (X_i, Y_i).$$

Risk

To measure the error (or loss) we make on a data point, define a function $\ell(h; D_i)$,

$$\ell(h; D) = \sum_{i=1}^{n} \ell(h; D_i) \text{ where } D_i = (X_i, Y_i).$$

Risk

Define the **risk** of a hypothesis for the complete space of inputs and outputs as,

$$R(h) = \mathbb{E}_{\mathcal{X},\mathcal{Y}}[\ell(h; \mathcal{D})].$$

We define **empirical risk** of a hypothesis given data D as,

$$R_{\text{emp}}(h; D) = \ell(h; D).$$

Issues with ERM

The minimiser of sample risk is not necessarily the minimiser of true risk.

We can restrict the learning algorithm's ability to fully minimise empirical risk through **regularisation**.

Issues with ERM

The minimiser of sample risk is not necessarily the minimiser of true risk.

We can restrict the learning algorithm's ability to fully minimise empirical risk through **regularisation**.

Regularised Empirical Risk

To restrict the hypothesis space, we define the formulation of **regularised empirical risk** as,

$$R_{\text{reg}}(h; D) = R_{\text{emp}}(h; D) + \lambda\pi(\theta).$$

$\pi : \mathbb{R}^p \rightarrow \mathbb{R}$ is a regulariser and $\lambda \in \mathbb{R}^+$ is controls the strength of regularisation.

Cross Validation

To avoid “overfitting”, we need to define an estimate of true risk from empirical data.

Cross Validation

To avoid "overfitting", we need to define an estimate of true risk from empirical data.

The most popular way to do this is through Cross Validation (CV), where we break the observed data into small subsets to run multiple "validation" experiments.

[1]Arlot and Celisse (2010)

Cross Validation

To avoid “overfitting”, we need to define an estimate of true risk from empirical data.

The most popular way to do this is through Cross Validation (CV), where we break the observed data into small subsets to run multiple “validation” experiments.

One of the most effective methods for risk approximation is Leave One Out Cross Validation (LOOCV) [1]. This method is computationally expensive as we repeat the learning task n times.

[1] Arlot and Celisse (2010)

Section 2

Literature Review

Approximate CV Methods

To reduce the computational cost LOOCV, we turn to Approximate Cross Validation (ACV).

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.

Methods for ACV

There are three main methods for ACV in the literature.

- 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.

Methods for ACV

There are three main methods for ACV in the literature.

- 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.

We present a short summary of the theory behind these approximations.

Newton Step

We can redefine the definition for (regularised) empirical risk for a LOOCV experiment excluding a point with index j,

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

Newton Step

We can redefine the definition for (regularised) empirical risk for a LOOCV experiment excluding a point with index j,

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

The Jacobian of this form is,

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

Therefore the Hessian becomes,

$$\nabla_\theta^2 R_{\mathsf{reg}}(\theta; D_{-j}) = H(\theta; D) - \nabla_\theta^2 \ell(\theta, D_j)$$

$$\text{where} \qquad H(\theta; D) = \nabla_\theta^2 \left(\sum_{i=1}^{n} \ell(\theta; D_i) \right) + \lambda \nabla_\theta^2 \pi(\theta)$$

Therefore the Hessian becomes,

$$\nabla^2_\theta R_{\text{reg}}(\theta; D_{-j}) = H(\theta; D) - \nabla^2_\theta \ell(\theta, D_j)$$

$$\text{where} \quad H(\theta; D) = \nabla^2_\theta \left(\sum_{i=1}^{n} \ell(\theta; D_i) \right) + \lambda \nabla^2_\theta \pi(\theta)$$

Note that we assume $\ell(\cdot)$ and $\pi(\cdot)$ are both continuous and twice-differentiable functions.

Therefore the Hessian becomes,

$$\nabla^2_\theta R_{\text{reg}}(\theta; D_{-j}) = H(\theta; D) - \nabla^2_\theta \ell(\theta, D_j)$$

$$\text{where} \quad H(\theta; D) = \nabla^2_\theta \left(\sum_{i=1}^{n} \ell(\theta; D_i) \right) + \lambda \nabla^2_\theta \pi(\theta)$$

Note that we assume $\ell(\cdot)$ and $\pi(\cdot)$ are both continuous and twice-differentiable functions.

Now we can apply Newton's method for optimisation to take a "step" towards the LOOCV iterate $\hat{\theta}_{-j}$ by starting at the learned parameter $\hat{\theta}$.

We define the approximation of a LOOCV iterate as $\tilde{\theta}_{-j}$, where

$$\begin{aligned}\tilde{\theta}_{-j} &= \hat{\theta} - \left(H(\hat{\theta}; D) - \nabla_{\theta}^{2}\ell(\hat{\theta}; D_j)\right)^{-1}\left(\nabla_{\theta} R_{\mathsf{reg}}(\hat{\theta}; D) - \nabla_{\theta} R_{\mathsf{reg}}(\hat{\theta}; D_j)\right) \\ &= \hat{\theta} + \left(H(\hat{\theta}; D) - \nabla_{\theta}^{2}\ell(\hat{\theta}; D_j)\right)^{-1}\nabla_{\theta} R_{\mathsf{reg}}(\hat{\theta}; D_j).\end{aligned}$$

We define the approximation of a LOOCV iterate as $\tilde{\theta}_{-j}$, where

$$\begin{aligned}\tilde{\theta}_{-j} &= \hat{\theta} - \left(H(\hat{\theta}; D) - \nabla^2_\theta \ell(\hat{\theta}; D_j)\right)^{-1} \left(\nabla_\theta R_{\text{reg}}(\hat{\theta}; D) - \nabla_\theta R_{\text{reg}}(\hat{\theta}; D_j)\right) \\ &= \hat{\theta} + \left(H(\hat{\theta}; D) - \nabla^2_\theta \ell(\hat{\theta}; D_j)\right)^{-1} \nabla_\theta R_{\text{reg}}(\hat{\theta}; D_j).\end{aligned}$$

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

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

The quality of the approximation depends heavily on the fact that $\hat{\theta} \approx \theta^*$.

Infinitesimal Jackknife

We omit the derivation of IJ for brevity. The general idea is again to perform a first-order Taylor expansion to approximate LOOCV, though around the weights of a Jackknife.

The final form derived for this case is

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

with the same assumptions as in NS (loss and regularisation are continuously twice-differentiable, H is invertible and $\hat{\theta} \approx \theta^*$).

Iterative Approximate Cross Validation

Recently proposed, Iterative Approximate Cross Validation (IACV) and improves the existing methods by relaxing assumptions required for accuracy.

We solve the main learning task through an iterative method, where the updates are

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

for $S_k \subseteq [n]$ as a subset of indices and α_k is a learning rate at iteration k.

Iterative Approximate Cross Validation

Recently proposed, Iterative Approximate Cross Validation (IACV) and improves the existing methods by relaxing assumptions required for accuracy.

We solve the main learning task through an iterative method, where the updates are

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

for $S_k \subseteq [n]$ as a subset of indices and α_k is a learning rate at iteration k.

The explicit LOOCV update excluding a point i is defined as,

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

this step is what we aim to approximate.

The burden is in calculating the Jacobian $\nabla_\theta R_{\text{reg}}(\hat{\theta}_{-i}^{(k-1)}; D_{S_t \setminus i})$ for n points.

The burden is in calculating the Jacobian $\nabla_\theta R_{\text{reg}}(\hat{\theta}_{-i}^{(k-1)}; D_{S_t \setminus i})$ for n points. As an approximation, we use a second-order expansion of the Jacobian for $\hat{\theta}_{-i}^{(k-1)}$ centered around the estimate $\hat{\theta}^{(k-1)}$.

The burden is in calculating the Jacobian $\nabla_\theta R_{\text{reg}}(\hat{\theta}_{-i}^{(k-1)}; D_{S_t \setminus i})$ for n points. As an approximation, we use a second-order expansion of the Jacobian for $\hat{\theta}_{-i}^{(k-1)}$ centered around the estimate $\hat{\theta}^{(k-1)}$. Here,

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

is the estimate for the Jacobian.

The burden is in calculating the Jacobian $\nabla_\theta R_{\text{reg}}(\hat{\theta}_{-i}^{(k-1)}; D_{S_t \setminus i})$ for n points. As an approximation, we use a second-order expansion of the Jacobian for $\hat{\theta}_{-i}^{(k-1)}$ centered around the estimate $\hat{\theta}^{(k-1)}$. Here,

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

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,

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

It may seem counter-intuitive that we swap a simple Jacobian for a Jacobian *and* a Hessian in the approximation step.

[2] Y. Luo, Ren, and Barber (2023)

It may seem counter-intuitive that we swap a simple Jacobian for a Jacobian *and* a Hessian in the approximation step. 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	$n(A_p + B_p + D_p) + np^2$	$n^2 A_p + nD_p + np$

where A_p is one evaluation of the Jacobian, B_p is one evaluation of the Hessian, D_p is one evaluation of the proximal operator and K is the size of the subset used for SGD [2].

[2] Y. Luo, Ren, and Barber (2023)

It may seem counter-intuitive that we swap a simple Jacobian for a Jacobian *and* a Hessian in the approximation step. 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	$n(A_p + B_p + D_p) + np^2$	$n^2 A_p + nD_p + np$

where A_p is one evaluation of the Jacobian, B_p is one evaluation of the Hessian, D_p is one evaluation of the proximal operator and K is the size of the subset used for SGD [2].

We can however parallelise calculations while keeping $\hat{\theta}^{(k-1)}$ fixed for a measurable speedup.

[2] Y. Luo, Ren, and Barber (2023)

Define metrics for measuring ACV error,

$$\mathsf{Err}_{\mathsf{Approx}} = \frac{1}{n}\sum_{i=1}^{n} \|\tilde{\theta}_{-i}^{(k)} - \hat{\theta}_{-i}^{(k)}\|_2^2$$

Define metrics for measuring ACV error,

$$\mathsf{Err}_{\mathsf{Approx}} = \frac{1}{n}\sum_{i=1}^{n} \|\tilde{\theta}_{-i}^{(k)} - \hat{\theta}_{-i}^{(k)}\|_2^2$$

$$\mathsf{Err}_{\mathsf{CV}} = \frac{1}{n}\sum_{i=1}^{n} \left|\ell\left(\tilde{\theta}_{-i}^{(k)}; D_i\right) - \ell\left(\hat{\theta}_{-i}^{(k)}; D_i\right)\right|.$$

Define metrics for measuring ACV error,

$$\mathsf{Err}_{\mathsf{Approx}} = \frac{1}{n}\sum_{i=1}^{n} \|\tilde{\theta}_{-i}^{(k)} - \hat{\theta}_{-i}^{(k)}\|_2^2$$

$$\mathsf{Err}_{\mathsf{CV}} = \frac{1}{n}\sum_{i=1}^{n} \left| \ell\left(\tilde{\theta}_{-i}^{(k)}; D_i\right) - \ell\left(\hat{\theta}_{-i}^{(k)}; D_i\right) \right|.$$

The standard dataset $D = \{(X_i, Y_i)\}_{i=1}^n$ for $X_i \in \mathbb{R}^p$ and $Y_i \in \{0, 1\}$ is generated by

$$Y_i \sim \text{Bernoulli}(1/(\exp\left(-X_i^T \theta^*\right) + 1))$$

for $\theta^* \in \mathbb{R}^p$ with 5 non-zero entries.

We recreate the experiment shown in the paper $(n = 250, p = 20)$ as a sanity check of the Python implementation.

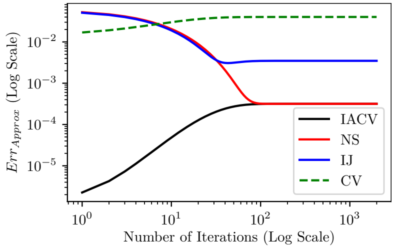


Figure 1: My implementation (run for less iterations).

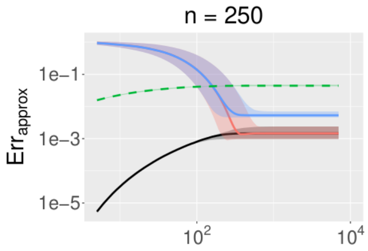


Figure 2: Experiment in the paper.

The 'baseline' we measure is using $\hat{\theta}^{(k)}$ as LOOCV estimates.

Gaps in the Literature

There are current gaps in the literature that we aim to address.

- Problems in high dimensions
- Lack of standard (or optimised) Python implementation of IACV
- No application of ACV to non GLM settings
- No attempt at applying ACV to models without a Hessian

Smooth Hinge Loss

The classic form of the hinge loss is not twice-differentiable, so we cannot apply ACV to any algorithm which uses it (SVM).

$$h(z) = \max\{0, 1 - z\}$$

Smooth Hinge Loss

The classic form of the hinge loss is not twice-differentiable, so we cannot apply ACV to any algorithm which uses it (SVM).

$$h(z) = \max\{0, 1 - z\}$$

We therefore seek a smooth approximation which is twice-differentiable to model the same problem an apply ACV for a fast approximation of true risk.

The paper (J. Luo, Qiao, and Zhang 2021), introduces a Smooth Hinge Loss of the form

$$\psi_M(z;\sigma) = \Phi_M(v)(1-z) + \phi_M(v)\sigma,$$

where $v = (1-z)/\sigma$ and $\sigma > 0$ controls the *smoothness* of the loss.

The paper (J. Luo, Qiao, and Zhang 2021), introduces a Smooth Hinge Loss of the form

$$\psi_M(z;\sigma) = \Phi_M(v)(1-z) + \phi_M(v)\sigma,$$

where $v = (1-z)/\sigma$ and $\sigma > 0$ controls the *smoothness* of the loss.

The individual functions which make up the loss are,

$$\begin{aligned}\Phi_M(v) &= \frac{1}{2}\left(1 + \frac{v}{\sqrt{1+v^2}}\right),\\ \phi_M(v) &= \frac{1}{2\sqrt{1+v^2}}.\end{aligned}$$

Add section on Smooth SVM problem itself and explain role of λ.

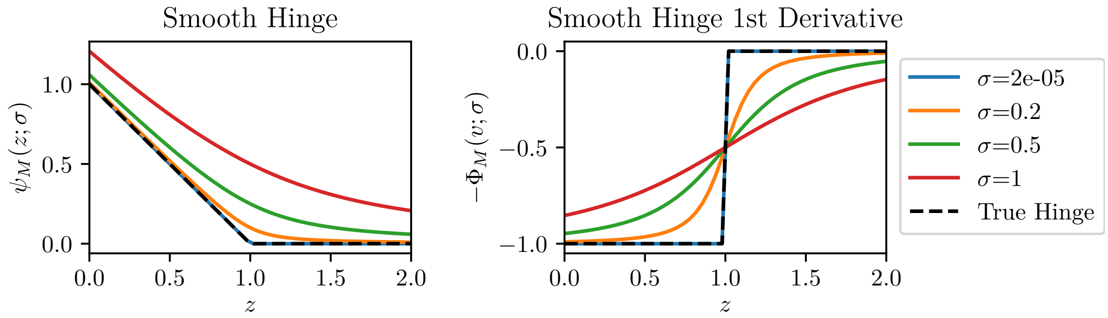


Figure 3: Smooth Hinge and its derivative for varying σ. Values for the original hinge loss are shown dotted.

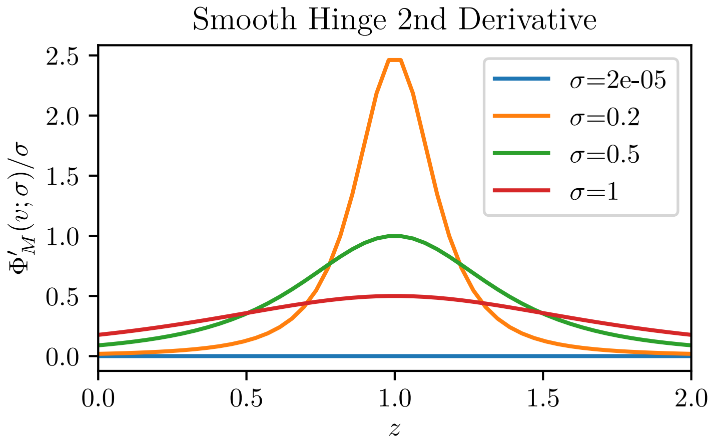


Figure 4: Second derivative of smooth hinge loss.

We can solve the Smooth SVM problem with GD and achieve results similar to solving SVM using QP (libsvm).

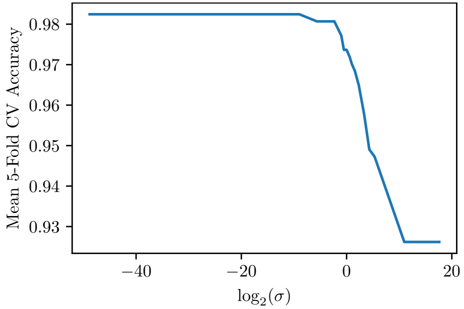


Figure 5: 5-Fold Accuracy of Smooth SVM solved using GD.

libsvm achieves a mean 97.88% accuracy on the same dataset (breast cancer binary classification from scikit-learn [3]).

[3] Pedregosa et al. (2011)

Section 3

Contributions

ACV Applied to Smooth Hinge

Previous attempts at applying ACV to problems without a Hessian are lacking. The Smooth SVM problem allows for ACV through its Hessian approximation.

ACV Applied to Smooth Hinge

Previous attempts at applying ACV to problems without a Hessian are lacking. The Smooth SVM problem allows for ACV through its Hessian approximation.

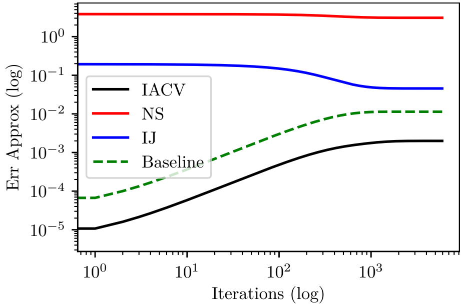


Figure 6: $\text{Err}_{\text{Approx}}$ for NS, IJ and IACV in the Smooth SVM problem ($\sigma = 0.5$ and $\lambda = 1$).

Why IACV?

- Less overall assumptions to satisfy.
- Accuracy before and at convergence. $\hat{w} \approx w^*$ not necessary.
- Per-iteration control of approximation.

Why IACV?

- Less overall assumptions to satisfy.
- Accuracy before and at convergence. $\hat{w} \approx w^*$ not necessary.
- Per-iteration control of approximation.

It's not that simple!

Main Result

IACV still needs the underlying problem to satisfy assumptions to guarantee accuracy above baseline.

- **Assumption 1:** the LOO Hessian is well-conditioned.

Main Result

IACV still needs the underlying problem to satisfy assumptions to guarantee accuracy above baseline.

- **Assumption 1:** the LOO Hessian is well-conditioned.
- **Assumption 2:** the LOO Jacobian is bounded along the convergence path.

Main Result

IACV still needs the underlying problem to satisfy assumptions to guarantee accuracy above baseline.

- **Assumption 1:** the LOO Hessian is well-conditioned.
- **Assumption 2:** the LOO Jacobian is bounded along the convergence path.
- **Assumption 3:** the LOO Hessian is $n\gamma-$Lipschitz.

Main Result

IACV still needs the underlying problem to satisfy assumptions to guarantee accuracy above baseline.

- **Assumption 1:** the LOO Hessian is well-conditioned.
- **Assumption 2:** the LOO Jacobian is bounded along the convergence path.
- **Assumption 3:** the LOO Hessian is $n\gamma-$Lipschitz.

The Smooth SVM problem satisfies Assumptions 2 and 3, though we need to find when Assumption 1 holds.

To investigate Assumption 1, we define the condition number. The LOO Hessian is provably positive-definite and symmetric, so we can use this bound.

Condition Number

For a positive-definite symmetric matrix $A \in \mathbb{R}^{n \times n}$, the condition number $\kappa(A)$ is defined as

$$\kappa(A) = \|A\|\|A^{-1}\| = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}.$$

We now present the main result.

Bound for LOO Hessian Condition Number

The condition number of the leave-one-out Hessian for the smooth SVM problem is bound by,

$$\kappa(\nabla_w^2 L(w; D_{-i}, \sigma)) \leq 1 + \frac{C}{\lambda\sigma} \cdot \frac{1}{2\sqrt{1 + (\frac{m_j}{\sigma})^2}^3},$$

for $C = \|\tilde{X}^T\tilde{X}\|/(n-1)$ where $m_j = (1 - Y_j w^T X_j)$ is derived from the term

$$\max_j d_j = \max_j \Phi_M'((1 - Y_j w^T X_j)/\sigma)/\sigma.$$

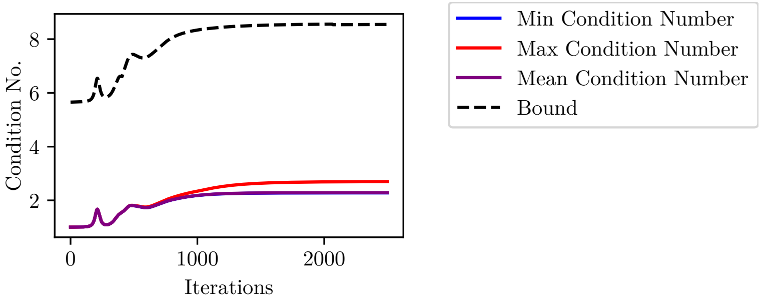


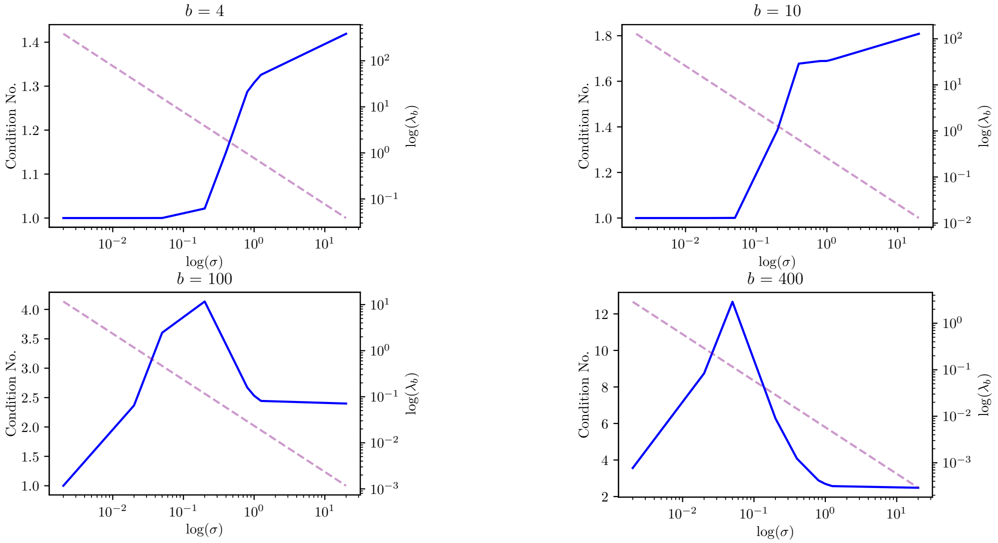
Figure 7: Bound along condition numbers across convergence path.

To use this bound, we can find an optimal λ for a given σ for a well-conditioned LOO Hessian.

To use this bound, we can find an optimal λ for a given σ for a well-conditioned LOO Hessian. For a chosen bound b and a given σ, the chosen λ must be at least

$$\lambda_b = \frac{C_f}{(b-1)\sigma} \cdot \frac{1}{2\sqrt{1 + \left(\frac{m^*}{\sigma}\right)^2}^3},$$

for a well-conditioned LOO Hessian. Here, $C_f = \|X^T X\|/(n-1)$ and m^* is a small constant close to 0.


$b = 4$
$b = 10$
$b = 100$
$b = 400$
Condition No.
$\log(\sigma)$
$\log(\lambda_b)$
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
10^{-3}
1.0
1.1
1.2
1.3
1.4
1.6
1.8
1.5
2.0
2.5
3.0
3.5
4.0
2
4
6
8
10
12

Does the bound work in improving IACV accuracy?

Does the bound work in improving IACV accuracy?

Yes! This is an experiment run for $\sigma = 1 \times 10^{-10}$, picking an optimisitic bound of $b = 1 \times 10^{10}$ when picking λ_b.

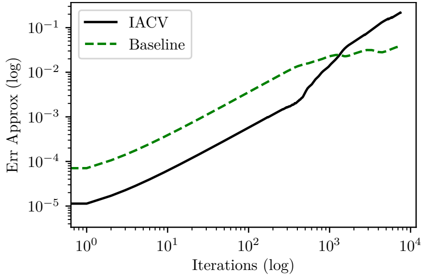


Figure 8: $\text{Err}_{\text{Approx}}$ when choosing $\lambda = 1$.

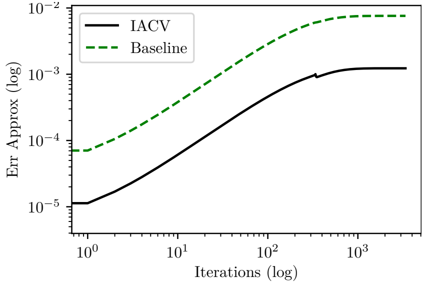


Figure 9: $\text{Err}_{\text{Approx}}$ when choosing $\lambda = \lambda_b$.

What do the condition numbers look like?

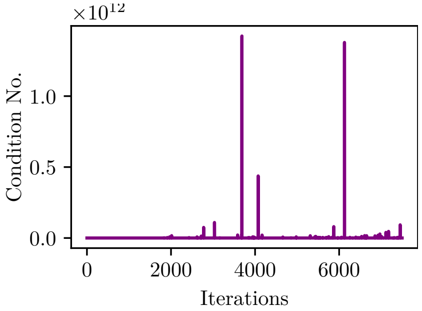


Figure 10: Condition number when choosing $\lambda = 1$.

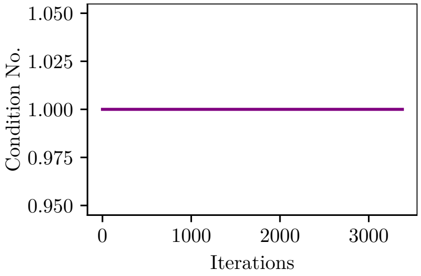


Figure 11: Condition number when choosing $\lambda = \lambda_b$.

Does choosing λ_b impact the overall quality of the model?

Does choosing λ_b impact the overall quality of the model?

Only slightly.

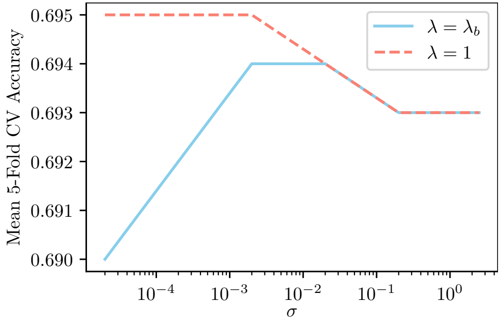


Figure 12: Accuracy for $\sigma = 0.1$ when picking $\lambda = 1$ and $\lambda = \lambda_b$.

To affirm the theory, we run a sensitivity study for both σ and λ. For the tests, we keep $\lambda = 1$ and $\sigma = 0.25$ fixed respectively.

To affirm the theory, we run a sensitivity study for both σ and λ. For the tests, we keep $\lambda = 1$ and $\sigma = 0.25$ fixed respectively.

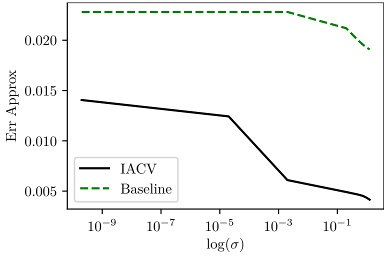


Figure 13: $\text{Err}_{\text{Approx}}$ for varying σ.

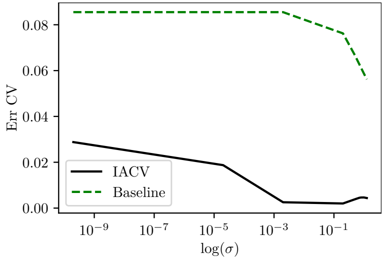


Figure 14: Err_{CV} for varying σ.

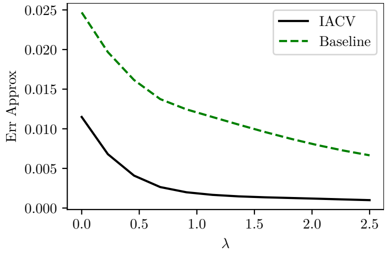


Figure 15: $\text{Err}_{\text{Approx}}$ for varying λ.

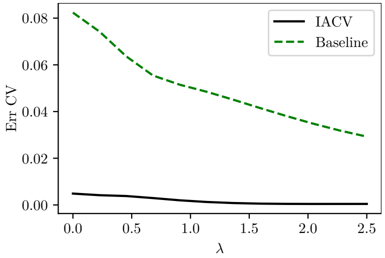


Figure 16: Err_{CV} for varying λ.

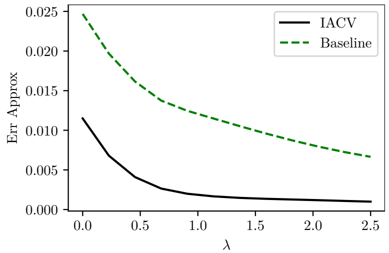


Figure 15: $\text{Err}_{\text{Approx}}$ for varying λ.

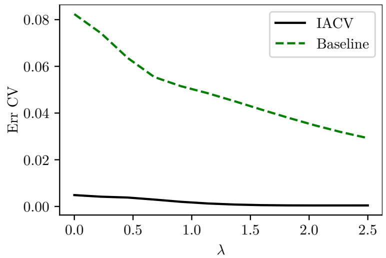
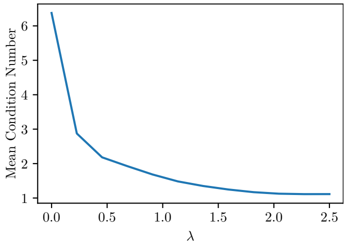
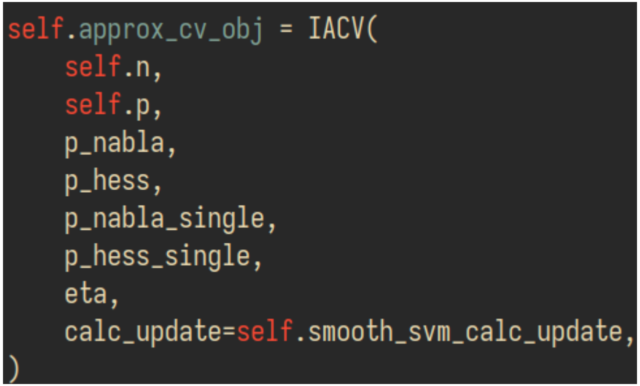


Figure 16: Err_{CV} for varying λ.



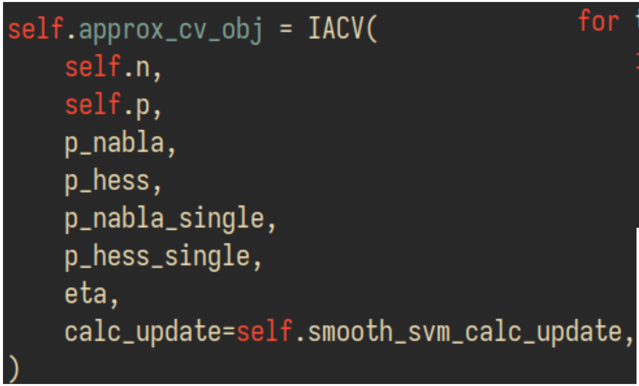
How do we use it?

How do we use it?

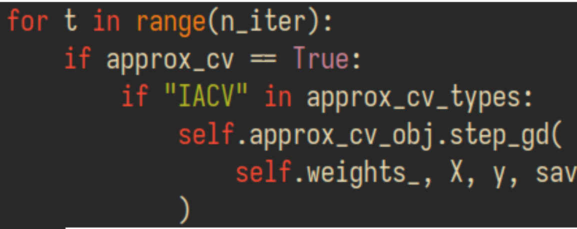


```
self.approx_cv_obj = IACV(
    self.n,
    self.p,
    p_nabla,
    p_hess,
    p_nabla_single,
    p_hess_single,
    eta,
    calc_update=self.smooth_svm_calc_update,
)
```

How do we use it?

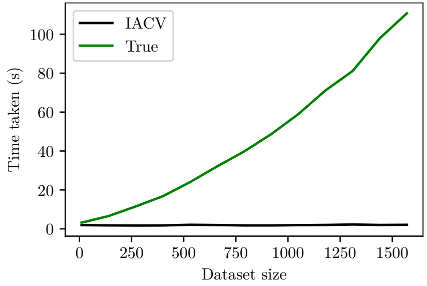


```
self.approx_cv_obj = IACV(
    self.n,
    self.p,
    p_nabla,
    p_hess,
    p_nabla_single,
    p_hess_single,
    eta,
    calc_update=self.smooth_svm_calc_update,
)
```



```
for t in range(n_iter):
    if approx_cv == True:
        if "IACV" in approx_cv_types:
            self.approx_cv_obj.step_gd(
                self.weights_, X, y, sav
            )
```

The underlying implementation uses JAX, making liberal use of vmap and jit to speed up runtime.



- ☒ why IACV? show NS and IJ performing poorly
- ☒ main result of bound
- ☒ show experiment of bound working
- ☒ show $\lambda = \lambda_b$
- ☒ show λ and σ sensitivity study
- ☐ computation contribution + pseudocode
- ☐ shorter section on kernel SVM (bound result shows more robustness + experiment)
- ☐ conclusion & further work?

Section 4

Bibliography

Bibliography

Arlot, and Celisse. 2010. "A Survey of Cross-Validation Procedures for Model Selection." *Statistics Surveys* 4 (40-79).

Luo, JunRu, Hong Qiao, and Bo Zhang. 2021. "Learning with Smooth Hinge Losses." https://arxiv.org/abs/2103.00233.

Luo, Yuetian, Zhimei Ren, and Rina Foygel Barber. 2023. "Iterative Approximate Cross-Validation." https://arxiv.org/abs/2303.02732.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. "Scikit-Learn: Machine Learning in Python." *Journal of Machine Learning Research* 12: 2825–30.