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,

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

$\pi : \mathbb{R}^p \to \mathbb{R}$ 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.

We can do this through Cross Validation (CV).

[1] Arlot and Celisse (2010)

Cross Validation

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

We can do this through Cross Validation (CV).

Leave One Out Cross Validation (LOOCV) is effective [1], but costly.

[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

In summary, we take a Newton Step on an approximation of the objective function.

Newton Step

In summary, we take a Newton Step on an approximation of the objective function.

Note that we assume $\ell(\cdot)$ and $\pi(\cdot)$ are both continuous and twice-differentiable functions, also that the Hessian is invertible.

Newton Step

In summary, we take a Newton Step on an approximation of the objective function.

Note that we assume $\ell(\cdot)$ and $\pi(\cdot)$ are both continuous and twice-differentiable functions, also that the Hessian is invertible.

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

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

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

Infinitesimal Jackknife

The general idea is again to perform a first-order Taylor expansion to approximate LOOCV, 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

Iterative Approximate Cross Validation (IACV) relaxes the assumptions of previous methods.

We solve the main learning task through updates,

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

for $S_k \subseteq [n]$ as a subset of indices and α_k.

Iterative Approximate Cross Validation

Iterative Approximate Cross Validation (IACV) relaxes the assumptions of previous methods.

We solve the main learning task through updates,

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

for $S_k \subseteq [n]$ as a subset of indices and α_k.

The LOO update 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.

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

The cost is in evaluating $\nabla_\theta R_{\text{reg}}(\hat{\theta}_{-i}^{(k-1)}; D_{S_t \setminus i})$ for n points. We use a Taylor expansion of the Jacobian for $\hat{\theta}_{-i}^{(k-1)}$ centered around the estimate $\hat{\theta}^{(k-1)}$ for approximation.

The cost is in evaluating $\nabla_\theta R_{\text{reg}}(\hat{\theta}_{-i}^{(k-1)}; D_{S_t \setminus i})$ for n points. We use a Taylor expansion of the Jacobian for $\hat{\theta}_{-i}^{(k-1)}$ centered around the estimate $\hat{\theta}^{(k-1)}$ for approximation. Here,

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

The cost is in evaluating $\nabla_\theta R_{\text{reg}}(\hat{\theta}_{-i}^{(k-1)}; D_{S_t \setminus i})$ for n points. We use a Taylor expansion of the Jacobian for $\hat{\theta}_{-i}^{(k-1)}$ centered around the estimate $\hat{\theta}^{(k-1)}$ for approximation. Here,

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

Therefore, the IACV updates for GD and SGD become,

$$\tilde{\theta}_{-i}^{(k)} = \tilde{\theta}_{-i}^{(k-1)} - \alpha_k \tilde{J}_{-i}^{(k)}$$

The cost is in evaluating $\nabla_\theta R_{\text{reg}}(\hat{\theta}_{-i}^{(k-1)}; D_{S_t \setminus i})$ for n points. We use a Taylor expansion of the Jacobian for $\hat{\theta}_{-i}^{(k-1)}$ centered around the estimate $\hat{\theta}^{(k-1)}$ for approximation. Here,

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

Therefore, the IACV updates for GD and SGD become,

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

We swap a Jacobian for a Jacobian *and* a Hessian in the approximation step.

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

We swap a 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)

We swap a 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

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

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.

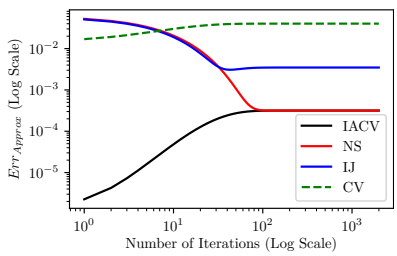


Figure 1: My implementation (run for less iterations).

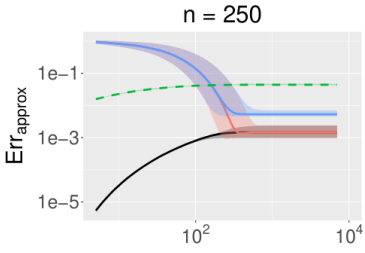


Figure 2: Experiment in the paper.

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,

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

We then use this loss to define the main objective,

$$L(w; D, \sigma) = \frac{\lambda}{2}\|w\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} \psi_M(Y_i w^T X_i; \sigma)$$

where λ controls the size of the margin.

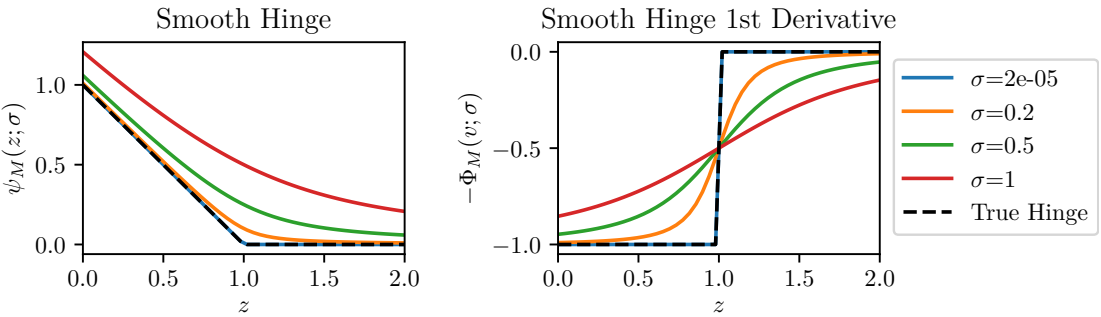


Figure 3: Smooth Hinge and its derivative for varying σ. Values for the original hinge loss are shown dotted.

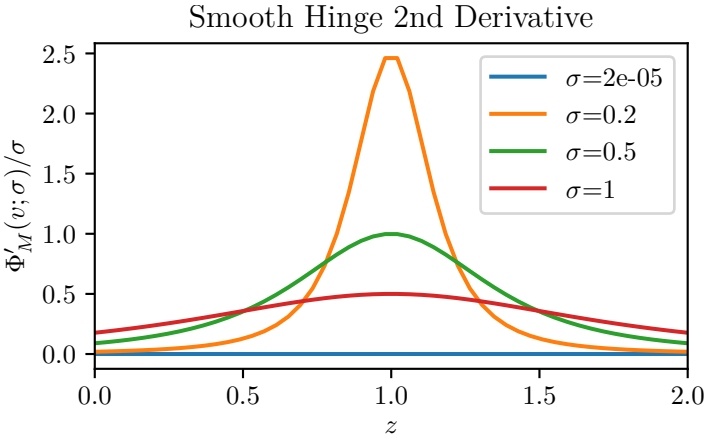


Figure 4: Second derivative of smooth hinge loss.

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

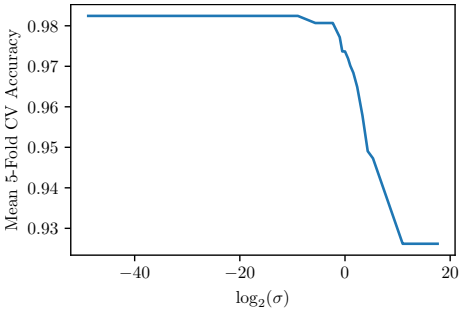


Figure 5: 5-Fold Accuracy of Smooth SVM solved using GD.

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.

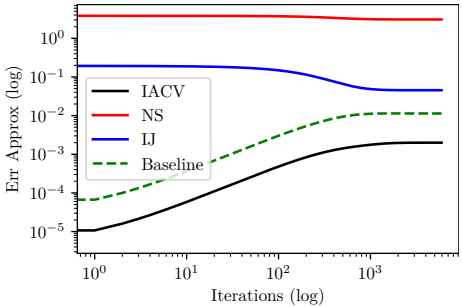


Figure 6: $\text{Err}_{\text{Approx}}$ for NS, IJ and IACV in the Smooth SVM problem ($\sigma = 0.5$ and $\lambda = 1$).

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,

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

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

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

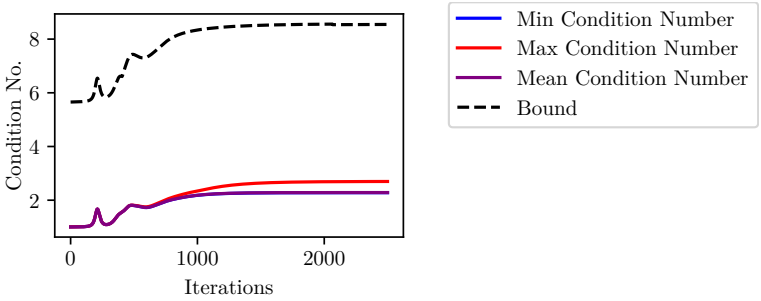


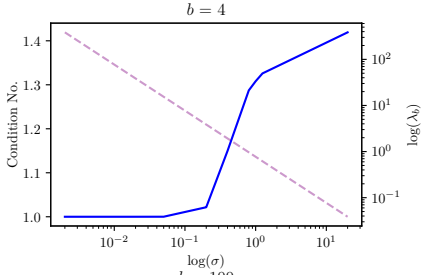
Figure 7: Bound along condition numbers across convergence path. ($\sigma = 0.1, \lambda = 1$)

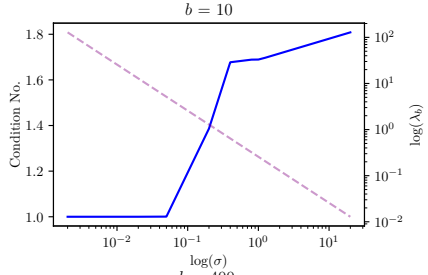
To use this bound, we can find an optimal λ for a given σ for a well-conditioned LOO Hessian.

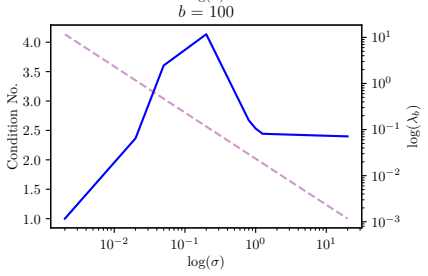
To use this bound, we can find an optimal λ for a given σ for a well-conditioned LOO Hessian. For a chosen bound b and a given σ, the chosen λ must be at least

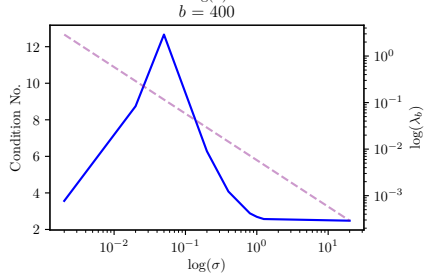
$$\lambda_b = \frac{C_f}{(b-1)\sigma} \cdot \frac{1}{2\sqrt{1 + \left(\frac{m^*}{\sigma}\right)^2}^3},$$

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


$b = 4$
Condition No.
$\log(\lambda_6)$
$\log(\sigma)$


$b = 10$
Condition No.
$\log(\lambda_6)$
$\log(\sigma)$


$b = 100$
Condition No.
$\log(\lambda_6)$
$\log(\sigma)$


$b = 400$
Condition No.
$\log(\lambda_6)$
$\log(\sigma)$

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 optimistic bound of $b = 1 \times 10^{10}$ when picking λ_b.

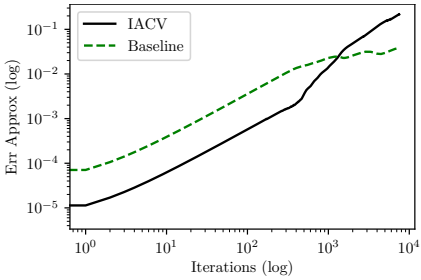


Figure 8: $\text{Err}_{\text{Approx}}$ when choosing $\lambda = 1$.

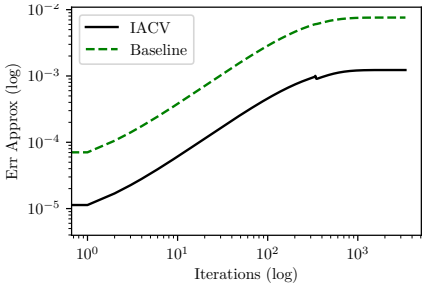


Figure 9: $\text{Err}_{\text{Approx}}$ when choosing $\lambda = \lambda_b$.

What do the condition numbers look like?

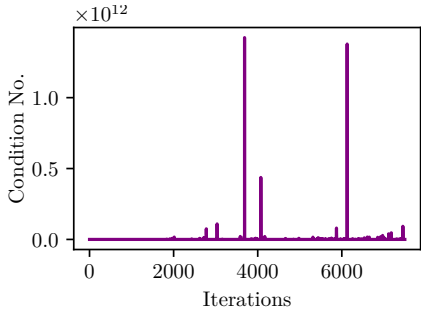


Figure 10: Condition number when choosing $\lambda = 1$.

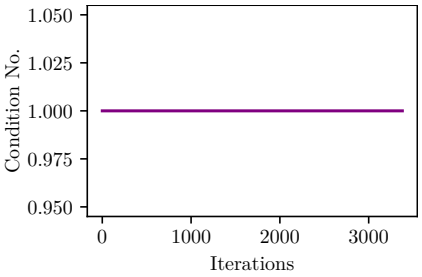


Figure 11: Condition number when choosing $\lambda = \lambda_b$.

Does choosing λ_b impact the overall quality of the model?

Does choosing λ_b impact the overall quality of the model?

Only slightly.

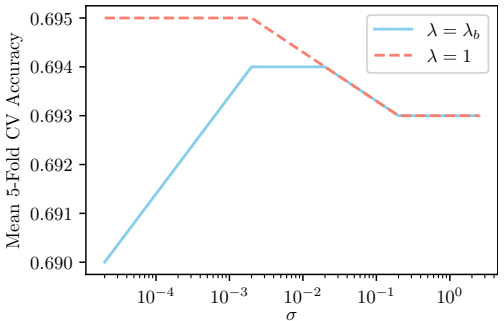


Figure 12: Accuracy for $\sigma = 0.1$ when picking $\lambda = 1$ and $\lambda = \lambda_b$.

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.

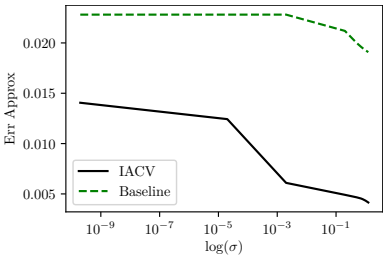


Figure 13: $\text{Err}_{\text{Approx}}$ for varying σ.

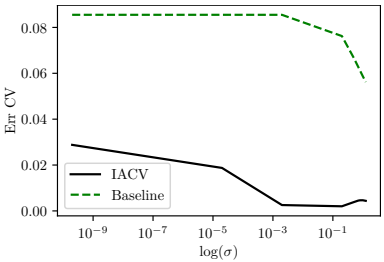


Figure 14: Err_{CV} for varying σ.

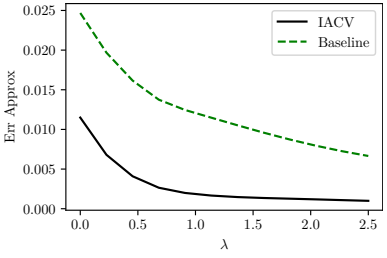


Figure 15: $\text{Err}_{\text{Approx}}$ for varying λ.

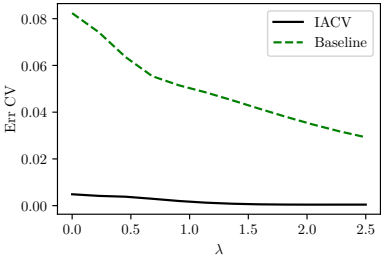


Figure 16: Err_{CV} for varying λ.

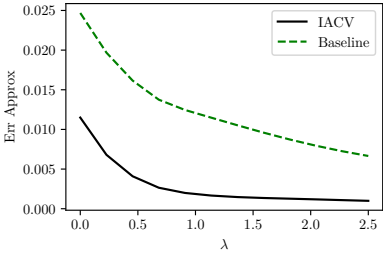


Figure 15: $\text{Err}_{\text{Approx}}$ for varying λ.

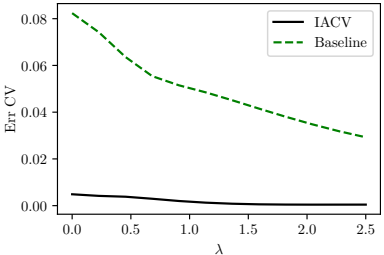
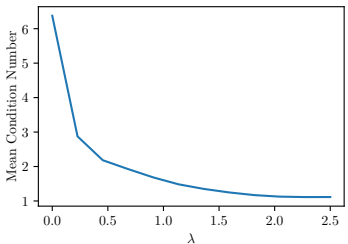


Figure 16: Err_{CV} for varying λ.



Python Implementation

How do we use it?

Python Implementation

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,
)
```

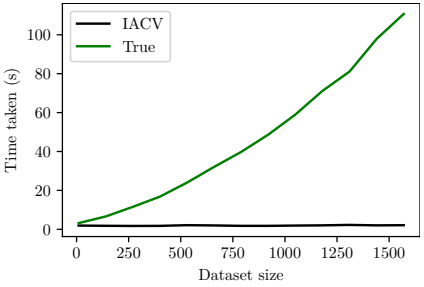
Python Implementation

How do we use it?

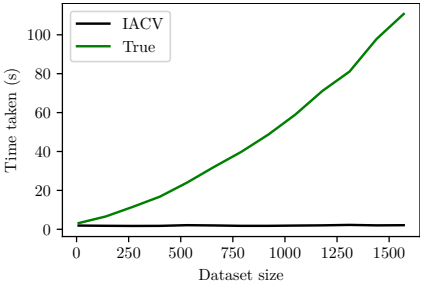
```
self.approx_cv_obj = IACV(
    self.n,
    self.p,
    p_nabla,
    p_hess,
    p_nabla_single,
    p_hess_single,
    eta,
    calc_update=self.smooth_svm_calc_update,
)
```

```
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.

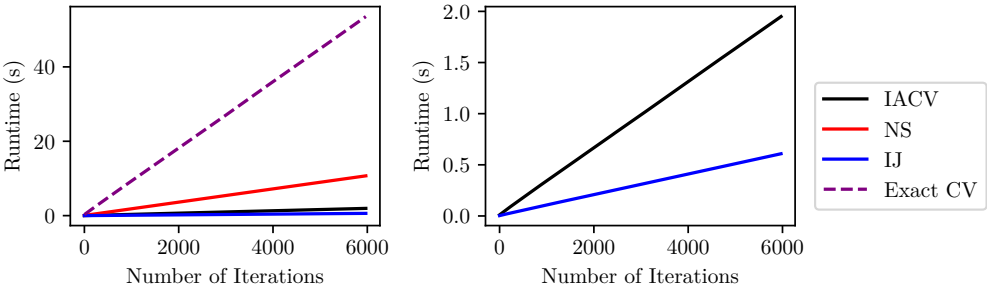


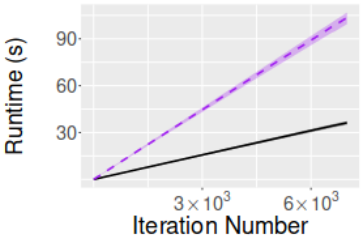
The underlying implementation uses JAX, making liberal use of vmap and jit to speed up runtime.



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

Improves on existing implementation by an order of magnitude.





Kernelising Smooth SVM

We can kernelise the smooth SVM by substituting $w = \phi(X) \cdot u$ for $u \in \mathbb{R}^n$,

$$L(u; D, \sigma) = \frac{\lambda}{2} u^T G u + \frac{1}{n} \sum_{i=1}^{n} \psi_M(Y_i G_i u; \sigma)$$

where G is the Gram matrix.

Kernelising Smooth SVM

We can kernelise the smooth SVM by substituting $w = \phi(X) \cdot u$ for $u \in \mathbb{R}^n$,

$$L(u; D, \sigma) = \frac{\lambda}{2} u^T G u + \frac{1}{n} \sum_{i=1}^{n} \psi_M(Y_i G_i u; \sigma)$$

where G is the Gram matrix.

This gives us non-linearity through the feature map $\phi : \mathbb{R}^p \to \mathbb{R}^d$ where $d > p$. **How does kernelisation impact IACV?**

The bound for the condition number in this case is,

$$\kappa(\nabla_u^2 L(u; D_{-i}, \sigma)) \leq \frac{\|\tilde{G}^{-1}\| \left(\lambda \|\tilde{G}\| + C_k B_k\right)}{\lambda}$$

where $C_k = \|\tilde{G}^T \tilde{G}\|/(n-1)$ and $B_k = 1/\left(2\sigma\sqrt{1 + (\frac{m^*}{\sigma})^2}^3\right)$.

The bound for the condition number in this case is,

$$\kappa(\nabla_u^2 L(u; D_{-i}, \sigma)) \leq \frac{\|\tilde{G}^{-1}\| \left(\lambda \|\tilde{G}\| + C_k B_k\right)}{\lambda}$$

where $C_k = \|\tilde{G}^T \tilde{G}\|/(n-1)$ and $B_k = 1/\left(2\sigma\sqrt{1 + (\frac{m^*}{\sigma})^2}^3\right)$.

Assumptions 2 and 3 hold similar to the linear SVM case.

The condition number is much less sensitive to change based on σ and depends mostly on λ.

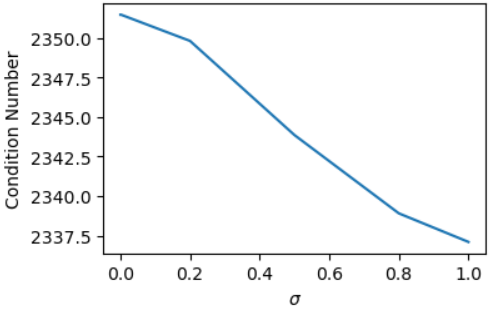


Figure 17: Kernel SVM.

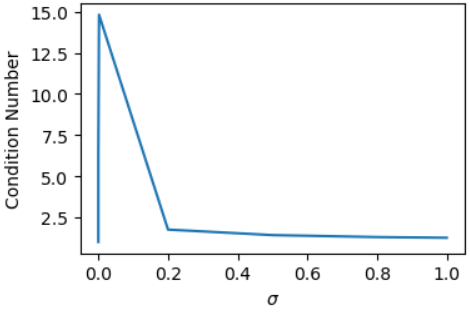


Figure 18: Linear SVM.

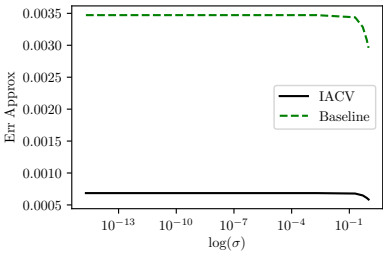


Figure 19: $\text{Err}_{\text{Approx}}$ for varying σ.

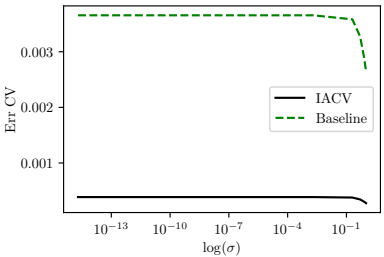


Figure 20: Err_{CV} for varying σ.

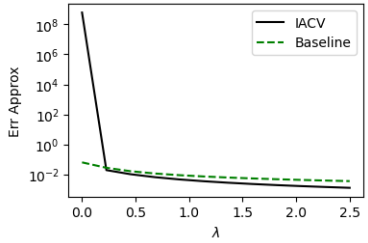


Figure 21: $\text{Err}_{\text{Approx}}$ for varying λ.

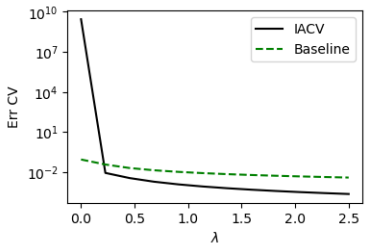
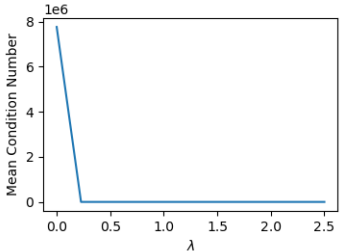


Figure 22: Err_{CV} for varying λ.



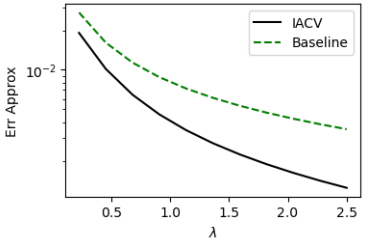


Figure 23: $\text{Err}_{\text{Approx}}$ for varying λ.

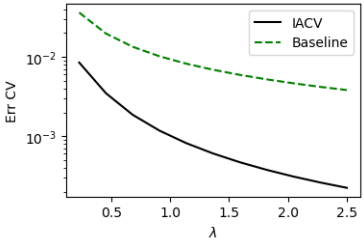
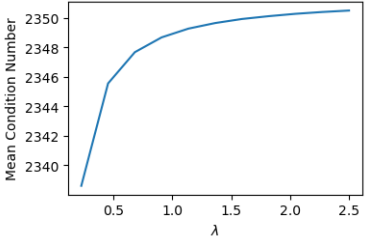


Figure 24: Err_{CV} for varying λ.



Section 4

Conclusion & Further Work

Conclusion & Further Work

Unfortunately, the original goal of applications to high dimensional problems was not fruitful.

In this thesis, we've explored ACV (specifically IACV) in a setting where an explicit Hessian does not exist without smoothing by applying it to Smooth SVM.

Conclusion & Further Work

Unfortunately, the original goal of applications to high dimensional problems was not fruitful.

In this thesis, we've explored ACV (specifically IACV) in a setting where an explicit Hessian does not exist without smoothing by applying it to Smooth SVM.

- Analysis of IACV on similar smoothed problems
  - Smoothed ReLU
- IACV in a non-convex setting
- Is IACV is high dimensions feasible?

Section 5

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.