Inverse Optimal Power Flow: Assessing the Vulnerability of Power Grid Data

Priya L. Donti,*[,1,2] Inês Lima Azevedo,[2] and J. Zico Kolter[1]

[1] School of Computer Science [2] Dept. of Engineering & Public Policy (Carnegie Mellon)

* pdonti@cmu.edu

We propose **inverse optimal power flow**, an algorithm to assess the extent to which private power grid data is compromised by public data. This algorithm **inverts the AC optimal power flow optimization problem** used to schedule electricity. Using this algorithm, we are able to learn private information such as electricity generation costs and (to some extent) grid structural parameters on a 14-bus test case.

Introduction

In the electricity sector, there is a great need to protect critical information that could compromise fair electricity market operation or power grid cybersecurity. At the same time, grid operators and governmental entities regularly publish grid data that could potentially expose this critical information.

Generator bidding curve parameters c

Grid structure Y

Nodal power demands p_d

Private data

$$\begin{aligned} \underset{z \equiv [\mathrm{angle}(v)^T \;\; |v|^T \;\; p_g^T \;\; q_g^T]}{\text{minimize}} \quad & f_c(p_g) \\ \text{subject to} \quad & Az = b \quad \text{(linear eq. constraints, with dual variable } \kappa) \\ & Gz \le h \quad \text{(linear ineq. constraints, with dual variable } \nu) \\ & (p_g - p_d) + qj = \mathrm{diag}(v)\bar{Y}\bar{v} \quad \text{(power flow constraint, with dual variable } \lambda) \end{aligned}$$

AC optimal power flow (ACOPF) relates private and public data

Generator power injections p_g

LMPs (prices) λ

Public data

Inverse optimal power flow algorithm

We assess the extent to which private grid data is exposed by public grid data by inverting ACOPF. Previous work has approached similar problems using techniques from game theory, graph theory, and bi-level optimization [1]. We formulate inverse optimal power flow (inverse OPF) as an optimization problem and solve it via gradient descent-based methods within a neural network.

Inverse OPF

$$\begin{aligned} \hat{c}^\star, \hat{Y}^\star = \underset{\hat{c},\hat{Y}}{\mathrm{argmin}} \quad & \ell\left((p_g,\lambda),(\hat{p}_g,\hat{\lambda})\right) \\ \text{subject to} \quad & \hat{p}_g, \hat{\lambda} = \mathrm{ACOPF}(\hat{c},\hat{Y},p_d) \end{aligned}$$

solve via gradient descent

input: $\left\{\left(p_g^{(i)},\lambda^{(i)}\right) \mid i = 1,\dots,m\right\}$ *// public data*

initialize: $\hat{c}, \hat{Y}$ *// some initial guess*

for $t = 1,\dots,T$:

  compute $\ell_{\mathrm{pub}} = \sum_{i=1}^{m} \ell\left(\left(p_g^{(i)},\lambda^{(i)}\right),\left(\hat{p}_g^{(i)},\hat{\lambda}^{(i)}\right)\right)$

  // update guesses if loss has not converged

  if $\ell_{\mathrm{pub}} \neq 0$ **then**

    update $\hat{c}$ **with** $\nabla_{\hat{c}}\ell_{\mathrm{pub}}$

    update $\hat{Y}$ **with** $\nabla_{\hat{Y}}\ell_{\mathrm{pub}}$

  else

    return $\hat{c}, \hat{Y}$

  end if

end for

Main challenge: Computing $\nabla_\theta \ell\left((p_g,\lambda),(\hat{p}_g,\hat{\lambda})\right)$ for each $\theta \in \{\hat{c},\hat{Y}\}$ (required for computing ℓ_{pub}). This involves gradients through the ACOPF solution since:

$$\frac{\delta \ell}{\delta \theta} = \frac{\delta \ell}{\delta \hat{p}_g}\frac{\delta \hat{p}_g}{\delta \theta} + \frac{\delta \ell}{\delta \hat{\lambda}}\frac{\delta \hat{\lambda}}{\delta \theta}.$$

To compute these gradients, we first solve ACOPF via sequential quadratic programming [2]. That is, we:

- assume $f_c(p_g) = p_g^T \mathrm{diag}(c_q) p_g + {c_a}^T p_g$ for quadratic and affine costs c_q, c_a,
- linearize the power flow constraint as $J(z_0)z = k(z_0)$ at some guess z_0,

and then solve the resultant QP iteratively (updating z_0) until the solution converges.

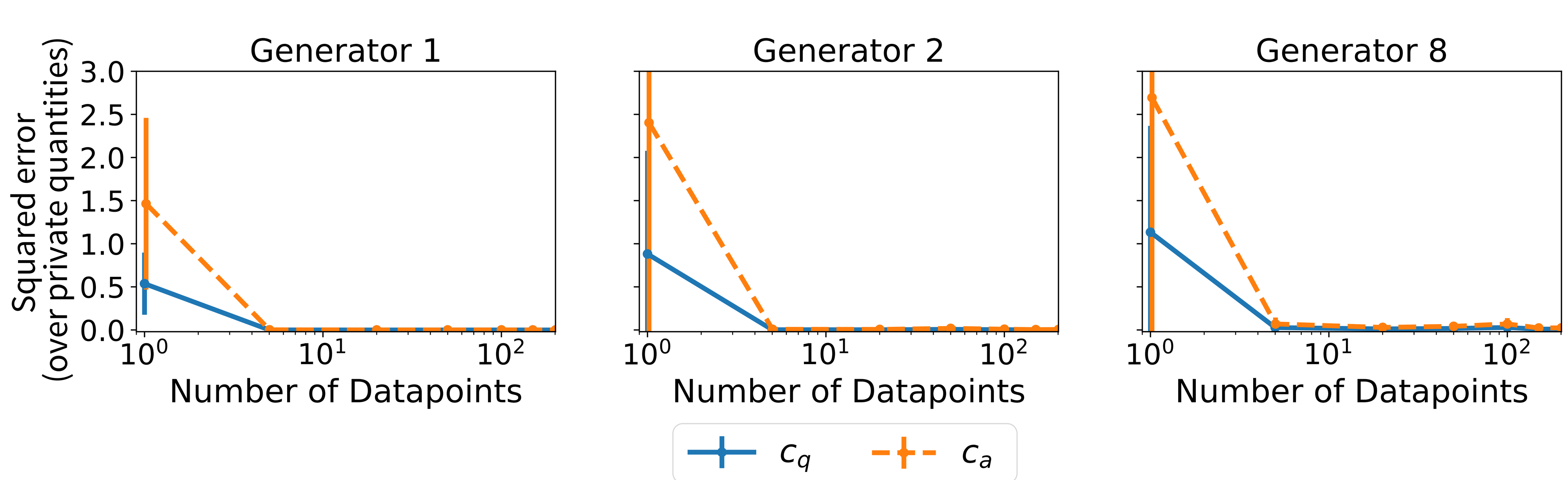
At the optimum, we implicitly differentiate the KKT conditions of the ACOPF QP relaxation, which yields a linear equation we can solve to efficiently get gradients [3]:

$$\begin{bmatrix} \mathrm{diag}(c_q) & G^T & \tilde{A}^T \\ \mathrm{diag}(\nu^\star)G & \mathrm{diag}(Gz^\star - h) & 0 \\ \tilde{A} & 0 & 0 \end{bmatrix} \begin{bmatrix} \partial z^\star/\partial\theta \\ \partial \nu^\star/\partial\theta \\ \partial \tilde{\kappa}^\star/\partial\theta \end{bmatrix} = \begin{bmatrix} -\frac{\partial \mathrm{diag}(c_q)}{\partial\theta} z^\star - \frac{\partial c_a}{\partial\theta} - \frac{\partial G^T}{\partial\theta}\nu^\star - \frac{\partial \tilde{A}^T}{\partial\theta}\tilde{\kappa}^\star \\ -\mathrm{diag}(\nu^\star)\left(\frac{\partial G}{\partial\theta} z^\star - \frac{\partial h}{\partial\theta}\right) \\ -\frac{\partial \tilde{A}}{\partial\theta} z^\star + \frac{\partial \tilde{b}}{\partial\theta} \end{bmatrix}$$

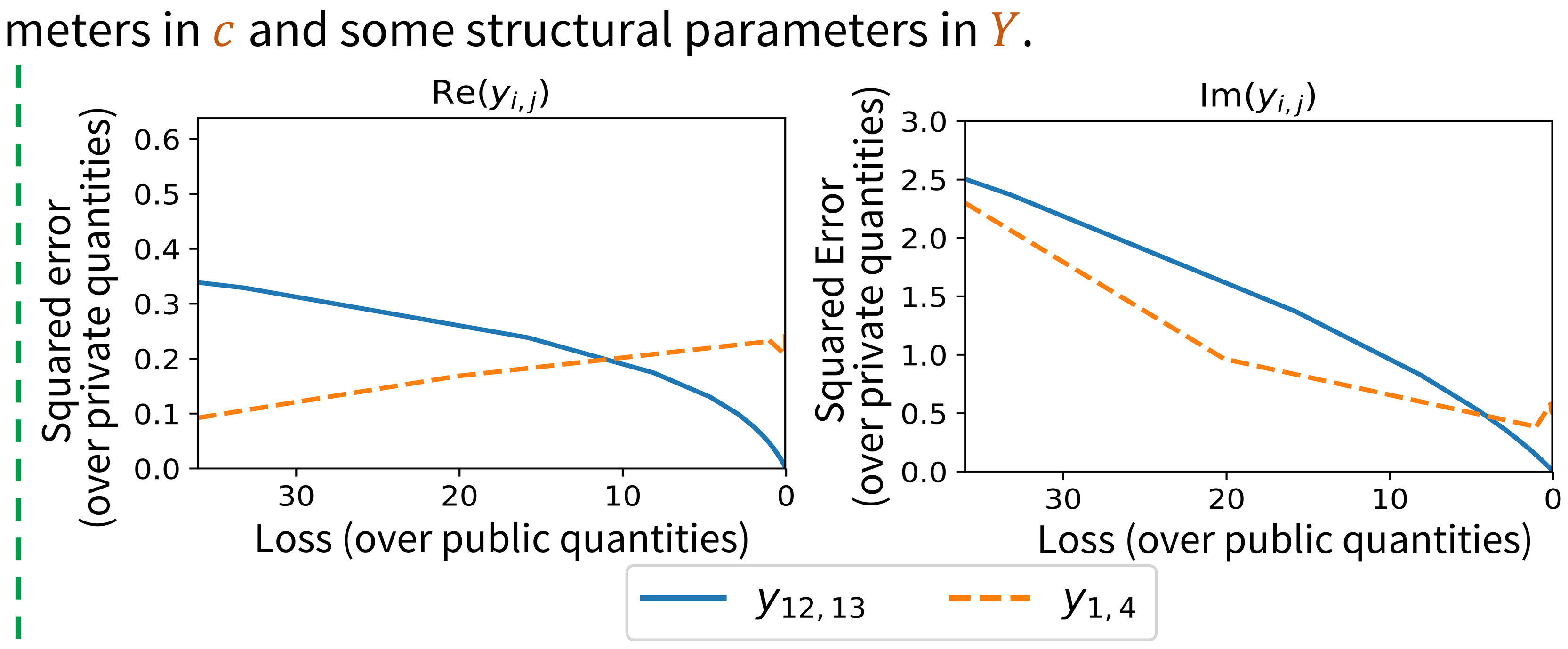
where $\tilde{A} = \begin{bmatrix} A \\ J(z^\star) \end{bmatrix}$, $\tilde{b} = \begin{bmatrix} b \\ k(z^\star) \end{bmatrix}$, and $\tilde{\kappa}^\star = \begin{bmatrix} \kappa^\star \\ \lambda^\star \end{bmatrix}$.

Preliminary results (14-bus system)

Preliminary experiments on a 14-bus system suggest that we are able to learn all cost parameters in c and some structural parameters in Y.



Squared error of final guesses for **generator cost parameters** in the scenario where all generators' costs are unknown (lower is better). Each plotted point represents five runs over a given amount of training data. We find that all cost parameters are identifiable with as little as 5 data points.



Squared error of **sample structural parameters** in the scenario that only one structural parameter is unknown. (Real and imaginary parts of each structural parameter are plotted separately.) Each plotted point represents the squared error of the unknown parameter as ℓ_{pub} goes to zero over all 201 data points used. Our estimate for $Y_{12,13}$ converges, but our estimate for $Y_{1,4}$ diverges.

[1] Yuan, Y. et al. (2016). On the Inverse Power Flow Problem. *arXiv preprint arXiv: 1610.06631.*

[2] Boggs, P. & Tolle, J. (1995). Sequential Quadratic Programming. *Acta Numerica 4:1-51.*

[3] Amos, B. & Kolter, J.Z. (2017). OptNet: Differentiable Optimization as a Layer in Neural Networks. *Proceedings of the 34th International Conference on Machine Learning, in PMLR* 70:136-145.

This work was supported by the Department of Energy Computational Science Graduate Fellowship under Grant No. DE-FG02-97ER25308.