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

Priya L. Donti  Inês Lima Azevedo  J. Zico Kolter
Dept. of Computer Science  Dept. of Engr. & Public Policy  Dept. of Computer Science
Pittsburgh, PA 15213  Pittsburgh, PA 15213  Pittsburgh, PA 15213
pdonti@cs.cmu.edu  iazevedo@cmu.edu  zkolter@cs.cmu.edu

Abstract

We formulate inverse optimal power flow, an algorithm that assesses the extent to which private power grid data is exposed by publicly-available data by inverting the AC optimal power flow optimization problem. We find that we are able to learn private information such as electricity generation costs and (to some extent) grid structural parameters on a 14-node test case.

1 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 [1, 2]. At the same time, grid operators such as PJM and governmental entities such as the Environmental Protection Agency regularly publish quantities such as five-minute electricity prices and hourly power outputs of electricity generators for the purposes of market transparency and emissions monitoring. We investigate the question of whether and to what extent critical power grid information is exposed by published information, given our knowledge that these private and public quantities are related via an optimization problem called AC optimal power flow (ACOPF). To do this, we formulate an algorithm called inverse optimal power flow (inverse OPF) that uses a neural network to learn private quantities from public quantities.

2 Related work

Inverse problems. Inverse problems seek to predict model inputs or decision parameters from model outputs. For instance, prior work has used techniques from game theory, graph theory, and bi-level optimization to identify power grid structure [3] and energy demands [4] from grid outputs. Our work seeks to apply techniques from optimization and deep learning to inverse power flow.

Differentiating through optimization problems. Recent work has explored differentiating through optimization problems within deep networks [5, 6]. We employ some of the innovations in differentiable quadratic programming to formulate and solve our inverse optimal power flow problem.

3 Background: AC optimal power flow

We now present the AC optimal power flow (ACOPF) optimization problem [7], which plays a crucial role in our formulation of inverse optimal power flow. ACOPF is solved by power system operators to pick power system quantities that minimize the overall cost of delivering power. Specifically, for a power grid with \( n \) nodes, operators must determine quantities \( z \equiv [\text{angle}(v)^T \ |v|^T \ p_g^T \ q_g]^T \).
where \( v \in \mathbb{C}^n \) are the voltages at each node and \( p_g, q_g \in \mathbb{R}^n \) are the real and imaginary parts of the power injections (e.g. from electricity generators) at all system nodes. These quantities solve

\[
\begin{align*}
\text{minimize} & \quad z = [\text{angle}(v)^T \ | v|^T \ p_g^T \ q_g^T]^T f_c(p_g) \\
\text{subject to} & \quad A z = b \quad \text{(linear equality constraints)} \\
& \quad G z \leq h \quad \text{(linear inequality constraints)} \\
& \quad (p_g - p_d) + (q_g - q_d) j = \text{diag}(v) \bar{Y} \bar{v} \quad \text{(power flow constraint)}.
\end{align*}
\]

Here, \( f_c : \mathbb{R}^n \rightarrow \mathbb{R} \) is a cost function parameterized by electricity generation costs \( c \); \( p_d, q_d \in \mathbb{R}^n \) are the real and imaginary power demands at all system nodes; \( \bar{Y} \in \mathbb{C}^{n \times n} \) is the nodal admittance matrix that describes how power flows throughout the system; the linear equality and inequality constraints encode attributes of system nodes, lines, and generators; and we use the notation \( \bar{x} \) to denote the complex conjugate of \( x \). We note that the dual variable \( \lambda \in \mathbb{R}^n \) on the power flow constraint corresponds to the electricity prices at each node.

In general, problem (1) is non-convex and NP-hard. As such, it is in practice common to assume that the objective is quadratic, linearize the power flow constraint using its Jacobian \( J \) at some point \( z_0 \) [7], and then solve the resulting problem iteratively via sequential quadratic programming [8]. Specifically, we write the linearized quadratic program corresponding to (1) as

\[
\begin{align*}
\text{minimize} & \quad z = [\text{angle}(v)^T \ | v|^T \ p_g^T \ q_g^T]^T p_g^T \ \text{diag}(c_q) p_g + c_a^T p_g \\
\text{subject to} & \quad \tilde{A} z = \tilde{b} \quad \text{(linear equality constraints)} \\
& \quad G z \leq h, \quad \text{(linear inequality constraints)}
\end{align*}
\]

where \( \tilde{A} = [A^T \quad J(z_0)^T]^T \) and \( \tilde{b} = [b^T \quad k(z_0)^T]^T \) collect both the original linear constraints and the linearized power flow constraint \( J(z_0) z = k(z_0) \), and where \( c = [c_q^T \quad c_a^T]^T \) now contains the quadratic and linear cost parameters \( c_q, c_a \in \mathbb{R}^n \), respectively, for power generation at each node.

### 4 Inverse optimal power flow

We now describe our inverse optimal power flow algorithm, which attempts to learn private electric within a neural network to compute the needed gradients. We note that while (3) maximizes the agreement between true and estimated public quantities, the objective of actual interest is the agreement between the true and estimated private quantities. However, there are potentially multiple distinct sets of inputs to ACOPF that would produce identical public outputs. Thus, we must use enough data when executing Algorithm [1] to ensure that there is a unique set of private parameters that can produce the correct public outputs across all input data points.

#### 4.1 Optimizing the inverse OPF problem

[2]

---

[1]: The reference number is used to denote the specific section or equation in the document, where the content is being referred to.

[2]: This section provides a detailed explanation of the inverse optimal power flow (OPF) algorithm, including its formulation and optimization. The approach involves using a neural network to learn private electric quantities that are not directly observable. The goal is to compute the needed gradients for optimization, which can be challenging due to the non-convex and NP-hard nature of the problem.

[3]: The method described in this section aims to optimize the inverse OPF problem, which is a critical component in power system analysis and control. The focus is on accurately estimating private parameters using available public data, ensuring that the solution is not only efficient but also secure from potential vulnerabilities.

---
We test the scenario in which all electricity generation cost parameters are unknown (but the admittance matrix is known). There is cause to believe that publicly-available data may expose generator cost parameters on the actual power system as well.

The main technical challenge of this approach is in computing the gradients $\nabla_{\theta} \ell((p_g, \lambda), (\hat{p}_g, \hat{\lambda}))$ for each $\theta \in \{\hat{c}, \hat{Y}\}$, as this involves taking the gradient through the solutions to the ACOPF optimization problem. Specifically, we must compute the terms

$$\frac{\partial \ell}{\partial \theta} = \frac{\partial \ell}{\partial p_g(\theta)} \frac{\partial p_g(\theta)}{\partial \theta} + \frac{\partial \ell}{\partial \lambda(\theta)} \frac{\partial \lambda(\theta)}{\partial \theta},$$

(4)

where $\frac{\partial p_g(\theta)}{\partial \theta}$ and $\frac{\partial \lambda(\theta)}{\partial \theta}$ are the Jacobians of optimal primal and dual variables, respectively, in problem (1), with respect to our parameter estimate $\theta$ (and where we denote the dependence of $\hat{p}_g$ and $\hat{\lambda}$ on each $\theta$ explicitly). To compute these Jacobians, we use the method presented in (5) to take gradients through the optimal quadratic program solved during the last iteration of sequential quadratic programming. At a high level, this involves differentiating through the KKT optimality conditions of (2) and using the implicit function theorem to get a set of linear equations we can solve to get the necessary gradients. More details on this approach are described in Appendix A.

5 Experiments

We test our algorithm on a modified version of the IEEE 14-bus test case with three generators located at nodes 1, 2, and 8, respectively. More details about this system are included in Appendix B. We train our network on up to 201 public outputs generated from Grid Optimization (GO) Competition simulations (9). We use the loss function

$$\ell((p_g, \lambda), (\hat{p}_g, \hat{\lambda})) = 100||p_g - \hat{p}_g||^2 + ||\lambda - \hat{\lambda}||^2$$

for (1), where the weighting term adjusts for differences in magnitude between values of $p_g$ and $\lambda$.

5.1 Cost parameters

We test the scenario in which all electricity generation cost parameters are unknown (but the admittance matrix is known). Results for runs over different amounts of training data are shown in Figure 1, with initial guesses for each cost parameter sampled from a Gaussian distribution to encode market participants’ prior knowledge of cost distributions. We find that we are able to completely learn the cost parameters for this system with as little as 5 public data points. Even though our test system is small, given that real power grid data is published with hourly granularity (i.e. 8760 data points per year), there is cause to believe that publicly-available data may expose generator cost parameters on the actual power system as well.
5.2 Admittance matrix parameters

![Admittance Matrix Parameters Diagram]

Admittance matrix parameters are potentially more challenging to learn than cost parameters, as the choice of admittance matrix parameters can potentially render problem (1) infeasible before or during training. As such, in our preliminary experiments, we test whether we can learn one admittance matrix parameter at a time, where our initial guess involves perturbing this parameter with Gaussian noise reflecting the variability across all admittance matrix parameters. (We assume all other cost and admittance matrix parameters are known.) As illustrated via representative results in Figure 2, our preliminary tests suggest that some admittance matrix parameters are readily identifiable while others may be harder to identify.

6 Conclusions and future work

We find that published power grid information can expose private information. Future work includes a more thorough investigation of admittance matrix parameters on the 14-node system, as well as assessments on larger systems. While we address power systems here, our method could be applied to any setting in which private and public information are related via a known optimization problem; extension of our method to other settings also remains as important future work.

Acknowledgments

This work is supported by the Department of Energy’s Computational Science Graduate Fellowship under grant number DE-FG02-97ER25308.

References


A Details on computing gradients through ACOPF

To compute the gradients \( \nabla_\theta \ell(\hat{p}_\theta^{(i)}, \lambda^{(i)}), (\hat{p}_\theta^{(i)}, \hat{\lambda}^{(i)}) ) \) for each \( \theta \in \{\hat{c}, \hat{Y}\} \), we must take the gradient through the solutions to the ACOPF optimization problem. Specifically, we must compute the terms

\[
\frac{\partial \ell}{\partial \theta} = \frac{\partial \ell}{\partial \hat{p}_g(\theta)} \frac{\partial \hat{p}_g(\theta)}{\partial \theta} + \frac{\partial \ell}{\partial \hat{\lambda}(\theta)} \frac{\partial \hat{\lambda}(\theta)}{\partial \theta}, \quad (A.1)
\]

where \( \frac{\partial \hat{p}_g(\theta)}{\partial \theta} \) and \( \frac{\partial \hat{\lambda}(\theta)}{\partial \theta} \) are the Jacobians of optimal primal and dual variables, respectively, in problem (1), with respect to our parameter estimate \( \theta \) (and where we denote the dependence of \( \hat{p}_g \) and \( \hat{\lambda} \) on each \( \theta \) here explicitly). To compute these Jacobians at the last sequential quadratic programming iterate, we use the method described in [5], implicitly differentiating through the KKT optimality conditions of (2) to obtain linear equations we can solve to obtain the required gradients:

\[
\begin{bmatrix}
\text{diag}(c_q) & G^T & \tilde{A}^T \\
\text{diag}(\nu^*)G & \text{diag}(G\tilde{z}^* - h) & 0 \\
\tilde{A} & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\frac{\partial \tilde{z}^*}{\partial \theta} \\
\frac{\partial \nu^*}{\partial \theta} \\
\frac{\partial \tilde{\kappa}^*}{\partial \theta}
\end{bmatrix}
= \begin{bmatrix}
- \frac{\partial \text{diag}(c_q)}{\partial \theta} \tilde{z}^* - \frac{\partial c_a}{\partial \theta} \tilde{z}^* - \frac{\partial G^T}{\partial \theta} \nu^* - \frac{\partial \tilde{A}^T}{\partial \theta} \tilde{\kappa}^* \\
- \text{diag}(\nu^*) \frac{\partial G}{\partial \theta} \tilde{z}^* + \text{diag}(\nu^*) \frac{\partial h}{\partial \theta} \\
- \frac{\partial \tilde{A}}{\partial \theta} \tilde{z}^* + \frac{\partial \tilde{b}}{\partial \theta}
\end{bmatrix}, \quad (A.2)
\]

where \( \nu \) are the dual variables on the linear inequality constraints, and \( \tilde{\kappa} = [\kappa^T \lambda^T]^T \) contains the dual variables \( \kappa \) on the original linear inequality constraints and \( \lambda \) on the linearized power flow constraint in (1). Here, we note that \( \tilde{A} = [A^T \; J(z^*)^T]^T \) and \( \tilde{b} = [b^T \; k(z^*)^T]^T \). While this equation may look complex, fundamentally, the left side of this equation contains the generalized Jacobian of the KKT optimality conditions of our convex problem, and the terms on the right side are the gradients of optimization problem parameters.

In practice, we solve a slightly different set of equations to efficiently compute these gradients in the context of a neural network, similar to the method described in [5]. In particular, we modify Equation (7) of [5] to incorporate the gradients of the loss with respect to both the optimal primal variable and the optimal dual variables on the equality constraints as

\[
\begin{bmatrix}
\frac{d\tilde{z}}{\partial \theta} \\
\frac{d\nu}{\partial \theta} \\
\frac{d\tilde{\kappa}}{\partial \theta}
\end{bmatrix}
= \begin{bmatrix}
\text{diag}(c_q) & G^T & \tilde{A}^T \\
G & \text{diag}(G\tilde{z}^* - h) & 0 \\
\tilde{A} & 0 & 0
\end{bmatrix}^{-1}
\begin{bmatrix}
\frac{\partial \ell}{\partial \tilde{z}}^T \\
\frac{\partial \ell}{\partial \nu}^T \\
\frac{\partial \ell}{\partial \tilde{\kappa}}^T
\end{bmatrix}. \quad (A.3)
\]

We then use the resultant values of \( d\tilde{z}, d\nu, \) and \( d\tilde{\kappa} \) in the rest of the computations presented in [5].
B Details on the 14-node test case

We test our algorithm on a modified version of the IEEE 14-bus test case with three generators located at nodes 1, 2, and 8, respectively. A schematic of this system is shown in Figure B.1. The power generation costs for each generator are $f_1(p_{g1}) = 2p_{g1}^2 + 5p_{g1}$, $f_2(p_{g2}) = 4p_{g2}^2 + 2p_{g2}$, and $f_8(p_{g8}) = 5p_{g8}^2 + 1p_{g8}$, and the primitive admittance matrix parameters used to construct the admittance matrix can be found at the link in Footnote 1.

![Figure B.1: The 14-node system on which we run our experiments.](https://www.cs.cmu.edu/~zkolter/course/15-884/assignments.html)