

# Implicit Differentiation

Justin Chiu  
Cornell Tech  
jtc257@cornell.edu

July 25, 2021

## Abstract

Gradient-based learning forms the foundation of modern machine learning, and automatic differentiation allows ML practitioners to easily compute gradients. While automatic differentiation only costs a constant multiple of the time and space required to evaluate a function, it has its limitations. In particular, when evaluating a function itself is expensive, the direct application of automatic differentiation is infeasible. In this report, we review the implicit function theorem (IFT) and its use in reducing the cost of computing gradients in scenarios where function evaluation is expensive, focusing on the application of implicit differentiation to variational inference.

## 1 Introduction

Gradient-based learning underpins many of the recent successes in machine learning, particularly advances involving neural networks. The key to the success of gradient-based methods is automatic differentiation (AD), which has greatly increased the development speed of machine learning research by allowing practitioners to circumvent the error-prone and time-consuming process of computing gradients manually. AD operates by reducing functions into compositions of atomic operations, for which we have a library of derivatives for, and composing those derivatives via the chain rule. (introduce the representation of functions as evaluation procedures / computational graphs, following Griewank and Walther [2008]) While efficient relative to the evaluation of the function in question, taking only a multiplicative constant longer than the evaluation itself, this may be prohibitively expensive if the original function evaluation itself is costly. An example of this is if the function takes the form of an unrolled loop, a common artifact of iterative methods. As naive AD requires storing all of the intermediate values at each point, storing the output of all computations at every iteration of a loop can quickly become infeasible due to memory limitations.

There are a variety of methods for overcoming the space limitations of AD, of which we only mention three: checkpointing, reversible computation, and implicit differentiation. A first method, checkpointing, improves space complexity at the cost of time. Rather than storing all intermediate computation, checkpointing instead recomputes values when needed. This can result in a large slowdown, and also requires careful choosing of which computational subgraphs to checkpoint. A

second method is an improvement upon checkpointing, called reversible computation [Maclaurin et al., 2015, Gomez et al., 2017], which improves space complexity at the cost of expressivity, but not speed. Reversible computation ensures that the gradient with respect to input depends only on the output, allowing the input to be discarded during function evaluation. This is typically accomplished by ensuring that the input is easily reconstructed from the output, restricting the expressivity of layers. A third method is implicit differentiation, which improves space complexity at the cost of stronger assumptions. Implicit differentiation relies on the implicit function theorem (IFT), which gives conditions under which derivatives can be computed independent of intermediate computation. Implicit differentiation requires a series of equations specified by a relation. In this report, we will cover the use of the implicit function theorem in OptNet [Amos and Kolter, 2017], which allows us to use the output of an optimization problem inside a neural network.

**Bilevel Optimization** One application of implicit differentiation is bilevel optimization. Bilevel optimization problems are, as implied by the name, optimization problems with another nested inner optimization problem embedded within. Methods for solving bilevel optimization typically proceed iteratively. For every iteration when solving the outer optimization problem, we must additionally solve an inner optimization problem. Some applications that can be formalized as bilevel optimization problems are hyperparameter optimization, metalearning, and variational inference.

Hyperparameter optimization formulates hyperparameter tuning, such as the shrinkage penalty in Lasso, as a bilevel optimization problem by computing gradients wrt the penalty through the entire learning procedure of the linear model [Lorraine et al., 2019]. (Other works on hyperparameter opt [Maclaurin et al., 2015, Bertrand et al., 2020]) Similarly, metalearning learns the parameters of a model such that the model is able to quickly be adapted to a new task via gradient descent [Finn et al., 2017, Rajeswaran et al., 2019]. This is accomplished by differentiating through the learning procedure of each new task. Finally, a variant of variational inference follows a very similar format: semi-amortized variational inference (SAVI) aims to learn a model that is able to initialize variational parameters [Kim et al., 2018]. This is also accomplished by differentiating through the iterative optimization procedure applied to the variational parameters during inference. (Other VI papers [Wainwright and Jordan, 2008, Johnson et al., 2017])

There is also work on expressing individual layers of a neural network declaratively as the solution of an optimization problem [Amos and Kolter, 2017, Agrawal et al., 2019, Gould et al., 2019]. This also falls under the umbrella of bilevel optimization, as we have both the outer training loop and the inner optimization loop for each OptNet layer.

## 2 The Implicit Function Theorem

The Implicit Function Theorem (IFT) has a long history, as well as many applications in a wide variety of fields such as economics and differential geometry. For an overview of the history of the IFT, see the book by Krantz and Parks [2003].

The IFT gives sufficient conditions under which the solution  $x$  to a system of equations,  $F(\theta, x) = 0$  with  $F : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^m$ , can locally be written as a function of just the parameters  $\theta$ , i.e. there

exists a solution mapping  $x^*$  such that  $f(\theta, x^*(\theta)) = 0$  in the neighbourhood of the particular point  $\theta \in \text{dom} F$ . These conditions are as follows:

1. We have a solution point  $(\theta, x)$  that satisfies the system of equations  $F(\theta, x) = 0$ .
2.  $F$  has at least continuous first derivatives:  $F \in \mathcal{C}^k$ .
3. The Jacobian of  $F$  wrt  $z$  evaluated at the solution point  $(\theta, x)$  is nonsingular:  $\det \frac{\partial F}{\partial x} \neq 0$ .

Given these conditions, we are able to assert the existence of the solution mapping  $x^*(\theta)$ , and determine its derivative  $\frac{\partial x^*(\theta)}{\partial \theta} = -[\frac{\partial F(\theta, x)}{\partial x}]^{-1} \frac{\partial F(\theta, x)}{\partial \theta}$ . Rather than directly applying the formula for the derivative, we can also use implicit differentiation to compute  $\frac{dx^*}{d\theta}$ .

While the IFT has a long history and many applications, we will focus on one particular application: We will use the solution to an optimization problem the output of a layer within a neural network, following OptNet [Amos and Kolter, 2017]. Afterwards, we will cover an application of the IFT to speed up variational inference.

### 3 Embedding Optimization inside a Neural Network

As a first example, we will replace the softmax layer of a neural network with an equivalent function defined as the output of an optimization problem, and derive derivatives using the IFT. Applying the IFT in this manner consists of three steps:

1. Check that the conditions of the IFT hold.
2. Compute the derivative of the implicit solution mapping wrt the parameters.

(Say check output and gradient) We will start by reviewing softmax and its expression as an optimization problem, checking the conditions of the IFT hold, then compute gradients. Since the gradient of softmax is known, we can easily verify that the IFT gives the correct gradients.

#### 3.1 Softmax

Softmax is often used to parameterize categorical distributions within neural networks, such as attention. It has its origins in statistical mechanics and decision theory, and functions as a differentiable surrogate for argmax.

Softmax assumes that we have  $n$  items with independent utilities,  $\theta \in \mathbb{R}^n$ , which indicate preferences. Softmax then gives the following distribution over these items:  $p(x) = \frac{\exp(\theta_x)}{\sum_y \exp(\theta_y)}$ . Interestingly, softmax arises as the solution of an optimization problem as well [Gao and Pavel, 2018].

The output of softmax is the solution of the following optimization problem:

$$\begin{aligned} \text{maximize} \quad & x^\top \theta + H(x) \\ \text{subject to} \quad & x^\top \mathbf{1} = 1 \\ & x \succeq 0, \end{aligned} \tag{1}$$

where  $H(x) = -\sum_i x_i \log x_i$  is the entropy. This corresponds to an entropy-regularized argmax optimization problem.

Our goal is to compute the Jacobian of softmax using the IFT and the optimization problem above. While this is not of practical use (there is a closed-form equation for both softmax and its Jacobian), we use it as an introduction to the mechanism behind OptNet and differentiable optimization layers [Amos and Kolter, 2017, Agrawal et al., 2019].

### 3.2 KKT Conditions

Given an optimization problem, the KKT conditions determine a system of equations that the solution must satisfy [Karush, 1939, Kuhn and Tucker, 1951]. First, we introduce dual variables  $u \in \mathbb{R}, v \in \mathbb{R}^n$  and write out the Lagrangian:

$$\mathcal{L}(\theta, x, u, v) = x^\top \theta + H(x) + u(x^\top \mathbf{1} - 1) + v^\top x.$$

We then have the following necessary conditions for a solution  $x$ :

$$\begin{aligned} \nabla_x \mathcal{L}(\theta, x, u, v) &= 0 && \text{(stationarity)} \\ u(x^\top \mathbf{1} - 1) &= 0 && \text{(primal feasibility)} \\ \text{diag}(v)x &= 0 && \text{(complementary slackness)} \\ v &\succeq 0 && \text{(dual feasibility)} \end{aligned} \tag{2}$$

As we are interested in a solution  $x$ , we focus on the first three conditions which give us the system of equations  $F(\theta, x)$  in the IFT.

In full, the system of equations  $F(\theta, x) = 0$  is

$$\begin{aligned} \theta + -\log(x) - 1 + u\mathbf{1} + v &= 0 \\ u(x^\top \mathbf{1} - 1) &= 0 \\ \text{diag}(v)x &= 0. \end{aligned} \tag{3}$$

Now we can check the conditions of the IFT. Clearly, any solution  $x$  must satisfy  $F(\theta, x) = 0$ , and  $F \in \mathcal{C}^1$ . We must then check that the Jacobian matrix  $\frac{dF}{dx}$  is non-singular.

Taking the differential of  $F(\theta, x, u, v) = 0$  yields

$$\begin{aligned} d\theta - \frac{dx}{x} + du\mathbf{1} + dv &= 0 \\ du(x^\top \mathbf{1} - 1) + u\mathbf{1}dx &= 0 \\ \text{diag}(v)dx + \text{diag}(x)dv &= 0. \end{aligned} \tag{4}$$

Rearranging into matrix form,

$$\begin{bmatrix} -\text{diag}(x)^{-1} & \mathbf{1} & I_n \\ u\mathbf{1}^\top & x^\top \mathbf{1} - 1 & 0 \\ \text{diag}(v) & 0 & \text{diag}(x) \end{bmatrix} \begin{bmatrix} dx \\ du \\ dv \end{bmatrix} = \begin{bmatrix} -d\theta \\ 0 \\ 0 \end{bmatrix} \tag{5}$$

Since the solution  $x^*$  must be feasible, we know that  $x^\top \mathbf{1} = 1$  and  $u > 0$ . With the additional information that the domain of  $H(x)$  adds the implicit constraint that  $\forall i, x_i > 0$ , we can deduce that the Jacobian is full rank and therefore has nonzero determinant. This shows that the conditions of the IFT hold.

### 3.3 The Jacobian of Softmax

Now that we have shown that the conditions of the IFT hold, we can proceed to compute the Jacobian of softmax by applying the second part of the IFT.

to the version of softmax expressed as an optimization problem in Eqn. 1. In particular, we are interested in the derivative  $\frac{dx}{d\theta}$ , the derivative of the output of softmax (obtained by solving the optimization problem) wrt the parameters  $\theta$ .

We can solve the system of equations in 3 for the entries of the upper-left block corresponding to  $\frac{dx}{d\theta}$ , i.e.

$$\begin{bmatrix} -\text{diag}(x)^{-1} & \mathbf{1} & I_n \\ u\mathbf{1}^\top & x^\top \mathbf{1} - 1 & 0 \\ \text{diag}(v) & 0 & \text{diag}(x) \end{bmatrix}^{-1} = \begin{bmatrix} \frac{dx}{d\theta} & \dots \\ \vdots & \ddots \end{bmatrix} \quad (6)$$

we compute the block-wise inverse of

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} (A - BD^{-1}C)^{-1} & 0 \\ 0 & (D - CA^{-1}B)^{-1} \end{bmatrix} \begin{bmatrix} I & -BD^{-1} \\ -CA^{-1} & I \end{bmatrix}$$

where

$$\begin{aligned} A &= \begin{bmatrix} -\text{diag}(x)^{-1} & \mathbf{1} \\ u\mathbf{1}^\top & 0 \end{bmatrix} & B &= \begin{bmatrix} I_n \\ 0 \end{bmatrix} \\ C &= [\text{diag}(v) & 0] & D &= \text{diag}(x) \end{aligned}$$

However, by complementary slackness we have  $v = 0$ , reducing the above to

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} & 0 \\ 0 & D^{-1} \end{bmatrix} \begin{bmatrix} I & -BD^{-1} \\ 0 & I \end{bmatrix}$$

We first compute the inverses:  $D^{-1} = \text{diag}(1/x)$ , and for  $A^{-1}$ , we use a similar block-wise inversion formula. Let us denote  $A = \begin{bmatrix} E & F \\ G & H \end{bmatrix}$ . First, we compute the Schur complement of  $A$ ,

$$A/E = H - GE^{-1}F = 0 - u\mathbf{1}^\top (-\text{diag}(x))\mathbf{1} = ux^\top \mathbf{1}. \quad (7)$$

Since  $x$  is feasible, we have  $A/E = u$ . Then, we have

$$A^{-1} = \begin{bmatrix} -\text{diag}(x)^{-1} & \mathbf{1} \\ u\mathbf{1}^\top & 0 \end{bmatrix}^{-1} = \begin{bmatrix} E & F \\ G & H \end{bmatrix}^{-1} = \begin{bmatrix} E^{-1} + E^{-1}F(A/E)^{-1}GE^{-1} & -E^{-1}F(A/E)^{-1} \\ -(A/E)^{-1}GE^{-1} & (A/E)^{-1} \end{bmatrix} \quad (8)$$

Plugging in, we have

$$\begin{aligned} A^{-1} &= \begin{bmatrix} -\text{diag}(x) + (-\text{diag}(x))\mathbf{1}u^{-1}u\mathbf{1}^\top(-\text{diag}(x)) & \text{diag}(x)\mathbf{1}u^{-1} \\ -u^{-1}u\mathbf{1}^\top(-\text{diag}(x)) & u^{-1} \end{bmatrix} \\ &= \begin{bmatrix} -\text{diag}(x) + xx^\top & u^{-1}x \\ x & u^{-1} \end{bmatrix} \end{aligned} \quad (9)$$

With this, we can finally compute the derivative  
as well as compute the derivative  $\frac{\partial x}{\partial \theta} = -[\frac{\partial F(\theta, x)}{\partial x}]^{-1} \frac{\partial F(\theta, x)}{\partial \theta}$ .

## 4 Semi-Amortized Variational Inference (POSTPONED)

Variational inference has found success in recent applications to generative models, in particular by allowing practitioners to depart from conjugate models and extend emission models with expressive neural network components. The main insight that led to this development is that inference can be amortized through the use of an inference network. One approach to variational inference, stochastic variational inference (SVI), introduces local, independent variational parameters for every instance of hidden variable. While flexible, the storage of all variational parameters is expensive, and the optimization of each parameter independently slow []. Amortized variational inference (AVI) solves that by instead sharing variational parameters hierarchically via an inference network, which in turn generates the local variational parameters []. The resulting local parameters may or may not be subsequently optimized.

Failure to further optimize may result in an amortization gap []. Prior work has shown that this gap can be ameliorated by performing a few steps of optimization on the generated local parameters obtained from the inference network, and even by propagating gradients through the optimization process. Optimizing through the inner optimization problem results in semi-amortized variational inference (SAVI) [].

As our main motivating example, we will examine whether we can apply the IFT to SAVI. We will start by formalizing the problem of variational inference for a simple model.

We will start with a model defined by the following generative process, used by Dai et al. [2019] to analyze posterior collapse:

1. Choose a latent code from the prior distribution  $z \sim p(z) = N(0, I)$ .
2. Given the code, choose an observation from the emission distribution  $x \mid z \sim p_\theta(x \mid z) = N(\mu_x(z, \theta), \gamma I)$ ,

where  $\mu_x(z, \theta) \equiv \text{MLP}(z, \theta)$  and  $\gamma > 0$  is a hyperparameter. This yields the joint distribution  $p(x, z) = p(x \mid z)p(z)$ .

Since the latent code  $z$  is unobserved, training this model would require optimizing the evidence  $p(x) = \int p(x, z)$ . However, due to the MLP parameterized  $\mu_x$ , the integral is intractable. Variational inference performs approximate inference by introducing variational distribution  $q_\phi(z \mid x)$

and maximizing the following lower bound on  $\log p(x)$ :

$$\log p(x) - D_{\text{KL}}[q(z|x)||p(z|x)] = \mathbb{E}_{q_\phi(z|x)} \left[ \log \frac{p_\theta(x, z)}{q_\phi(z|x)} \right] = \mathcal{L}(\theta, \phi). \quad (10)$$

(Write out objective in full.)

While SVI introduces local parameters for each instance of  $z$ , and AVI uses a single  $q(z|x)$  for all instances, we will follow the approach of SAVI. We will perform inference as follows: For each instance  $x$ , produce local variational parameter  $z^{(0)} = g(x; \phi)$ . Obtain  $z^*$  by solving  $\mathcal{L}(\theta, z^{(0)}) = 2$ , with (local) optima  $\ell^*$ . Take gradients through the whole procedure, i.e. compute  $\frac{\partial \ell^*}{\partial \phi} = \frac{\partial \ell^*}{\partial z^*} \frac{\partial z^*}{\partial z^{(0)}} \frac{\partial z^{(0)}}{\partial \lambda}$ . The main difficulty lies in computing  $\frac{\partial z^*}{\partial z^{(0)}}$ . (Highlight challenge)

In order to avoid the memory costs of storing all intermediate computation performed in a solver, we will instead apply the IFT. In order to apply the IFT, we must satisfy the three conditions. First, we must have a solution point to a system of equations,  $F(x_0, z_0) = 0$ . In this setting, we will use the KKT conditions of the optimization problem to define  $F$ .

## References

- A. Agrawal, B. Amos, S. T. Barratt, S. P. Boyd, S. Diamond, and J. Z. Kolter. Differentiable convex optimization layers. *CoRR*, abs/1910.12430, 2019. URL <http://arxiv.org/abs/1910.12430>.
- B. Amos and J. Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks. *CoRR*, abs/1703.00443, 2017. URL <http://arxiv.org/abs/1703.00443>.
- Q. Bertrand, Q. Klopfenstein, M. Blondel, S. Vaiter, A. Gramfort, and J. Salmon. Implicit differentiation of lasso-type models for hyperparameter optimization, 2020.
- B. Dai, Z. Wang, and D. P. Wipf. The usual suspects? reassessing blame for VAE posterior collapse. *CoRR*, abs/1912.10702, 2019. URL <http://arxiv.org/abs/1912.10702>.
- C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017. URL <http://arxiv.org/abs/1703.03400>.
- B. Gao and L. Pavel. On the properties of the softmax function with application in game theory and reinforcement learning, 2018.
- A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse. The reversible residual network: Backpropagation without storing activations. *CoRR*, abs/1707.04585, 2017. URL <http://arxiv.org/abs/1707.04585>.
- S. Gould, R. Hartley, and D. Campbell. Deep declarative networks: A new hope. *CoRR*, abs/1909.04866, 2019. URL <http://arxiv.org/abs/1909.04866>.
- A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, Philadelphia, 2nd edition, 2008.

- M. J. Johnson, D. Duvenaud, A. B. Wiltschko, S. R. Datta, and R. P. Adams. Composing graphical models with neural networks for structured representations and fast inference, 2017.
- W. Karush. *Minima of functions of several variables with inequalities as side conditions*. PhD thesis, Thesis (S.M.)—University of Chicago, Department of Mathematics, December 1939., 1939.
- Y. Kim, S. Wiseman, A. C. Miller, D. Sontag, and A. M. Rush. Semi-amortized variational autoencoders, 2018.
- S. Krantz and H. Parks. The implicit function theorem : History, theory, and applications / s.g. krantz, h.r. parks. 01 2003. doi: 10.1007/978-1-4612-0059-8.
- H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, 1950*, pages 481–492, Berkeley and Los Angeles, 1951. University of California Press.
- J. Lorraine, P. Vicol, and D. Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. *CoRR*, abs/1911.02590, 2019. URL <http://arxiv.org/abs/1911.02590>.
- D. Maclaurin, D. Duvenaud, and R. P. Adams. Gradient-based hyperparameter optimization through reversible learning, 2015.
- A. Rajeswaran, C. Finn, S. M. Kakade, and S. Levine. Meta-learning with implicit gradients. *CoRR*, abs/1909.04630, 2019. URL <http://arxiv.org/abs/1909.04630>.
- M. Wainwright and M. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1:1–305, 01 2008. doi: 10.1561/22000000001.

## A Example Appendix

Neural ODEs use reversibility.