Closed-form proximal operator of regularized exponential functions for incremental learning

Anonymous authors
Paper under double-blind review

Abstract

Incremental model-based minimization methods have recently been proposed as a way to mitigate numerical challenges associated with stochastic or online optimization. One of the main desirable properties is stability w.r.t. step-size choice and loss-function weights. Such properties make them desirable for use-cases when tuning parameters is prohibitive. In contrast to incremental gradient methods, the main computational tool is the proximal operator, rather than the gradient. And this operator is exactly one of the main gaps for adoption in practice - it may be both inefficient in practice, and harder to implement for a practitioner due to the lack of closed-form formulas and expressive calculus.

In this work, we aim to address this challenge for a specific family of losses, which are a composition of exponential on linear functions. One prominent application in mind is that of Poisson regression, where the negative log-likelihood is of this form. We devise a closed-form formula for the proximal operator in terms of Lambert's W function, whose implementation is available in many standard numerical computing and machine-learning packages, such as SciPy or TensorFlow. Then, we show that expressing the same formula in terms of the less-known Wright-Omega function, that is also available in SciPy, provides substantial numerical benefits. Finally, we provide an open-source vectorized PyTorch implementation of the Wright-Omega function and the proximal operator, ported from SciPy. This allows practitioners wishing to use the algorithm devised here to use the entire arsenal of tools provided by PyTorch, such as automatic differentiation and GPU computing. We have made our code available at `https://anonymous.4open.science/r/exponential-proximal-point-B8DD`.

1 Introduction

In modern machine learning, algorithms are used to *incrementally* minimize either the sum or expectation of functions of the form $h(\boldsymbol{w}, \boldsymbol{x})$, where the samples $\boldsymbol{x}$ come from a training set or are sampled from a distribution. Typically, the vector $\boldsymbol{w}$ denotes the parameters of the model we wish to learn, and h denotes the cost of mis-prediction. In both incremental paradigms, stochastic and online, the learning algorithm iteratively updates the current estimate of the parameters $\boldsymbol{w}$ based on the arriving samples $\boldsymbol{x}$.

The most popular incremental methods ranging from stochastic (Robbins & Monro, 1951) and online (Zinkevich, 2003; Gordon, 1999) gradient methods, AdaGrad (Duchi et al., 2011; McMahan & Streeter, 2010), AdamW (Loshchilov & Hutter, 2019), and others, use first-order information about the (sub) gradients of the function h w.r.t. $\boldsymbol{w}$. These methods, while attractive either theoretically or in practice, require careful tuning of their step-size, or learning rate. Wrong step-size selection may lead to sub-optimal performance at best, or even divergence and floating-point over-flow.

As an example, consider the problem of regularized Poisson regression (Nelder, 1974) for predicting the conditional mean $\mathbb{E}[y|\boldsymbol{x}] \sim \text{Poisson}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle)$ by minimizing the regularized negative log-likelihood over the training samples:

$$\frac{1}{m} \sum_{i=1}^{m} (\exp(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle) - y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle) + \frac{\alpha}{2} \|\boldsymbol{w}\|_2^2.$$

We immediately see that the summand gradients involve the exponential function, and therefore may be of a very large magnitude. With a step-size slightly too large, the model parameters will quickly grow in norm and cause a floating point overflow when we attempt to compute gradients for future training samples.

The recently devised model-based minimization methods, proposed by Asi & Duchi (2019); Davis & Drusvyatskiy (2019), are significantly more robust w.r.t. the step-size choice. Such algorithms are useful when either step-size tuning is overly expensive, or when the algorithm needs to run for a prolonged period of time without human supervision. These methods assume that ℓ is approximated by some model[1] f, which may depend on the current parameters $\boldsymbol{w}$ and the data point $\boldsymbol{x}$, and compute the updated parameters $\boldsymbol{w}^+$ using the *proximal operator* (Moreau, 1962; 1965):

$$\boldsymbol{w}^+ = \operatorname{prox}_{\eta f}(\boldsymbol{w}) \equiv \arg\min_{\boldsymbol{u}} \left\{ f(\boldsymbol{u}) + \frac{1}{2\eta}\|\boldsymbol{u} - \boldsymbol{w}\|_2^2 \right\}.$$

Intuitively, the idea is that our updated parameters balance minimizing the approximation of the cost, and staying close to the current parameters. This balance is determined by the step-size η.

When f is the Taylor approximation of the first order of ℓ, we obtain the incremental gradient method. Moreover, when the model is the cost function itself, namely, $f \equiv h$, we obtain the incremental proximal-point method. In this case, we use the structure of the entire cost function h rather than just its slope. Finally, in the vast majority of cases, the cost is a composition of a loss ℓ onto a machine-learned function m, namely, $h \equiv \ell \circ m$. In this case, we can construct a model of h by composing ℓ onto a first-order Taylor approximation of m. In this manner we obtain the Gauss-Newton or the prox-linear method. Note, that in this formulation m may be as complex as we desire, such as a neural network. These possible models, among others, were all studied in Asi & Duchi (2019); Davis & Drusvyatskiy (2019) and references therein.

In general, computing $\operatorname{prox}_{\eta f}$ heavily depends on the structure of f, and there is no efficient closed-form solution. Indeed, f may be as complex as we desire. Thus, the main challenge, both in terms of computational efficiency and usefulness in practice, lies in an easy to implement and efficient proximal operator.

In this work we do not propose yet another such algorithm, but rather deal with the implementation such algorithms for specific functions that appear in machine learning applications. We propose an implementation for model functions of the form:

$$f(\boldsymbol{w}; \boldsymbol{\theta}, \boldsymbol{\phi}, b, \alpha) = \exp(\langle \boldsymbol{\theta}, \boldsymbol{w} \rangle + b) + \langle \boldsymbol{\phi}, \boldsymbol{w} \rangle + \frac{\alpha}{2}\|\boldsymbol{w}\|_2^2. \tag{1}$$

By the notation $f(\boldsymbol{w}; \boldsymbol{\theta}, \boldsymbol{\phi}, b, \alpha)$ we mean a function of $\boldsymbol{w}$ parametrized by the remaining arguments. We use this notation since we study the minimization w.r.t. $\boldsymbol{w}$, and therefore we treat the remaining arguments as parameters. Our main application in mind is Poisson regression, where the regularized negative log-likelihood function w.r.t. each sample is a special case of this form. Nonetheless, additional applications exist, such as minimizing the exponentially tilted loss (Li et al., 2023), by transforming the minimization problem to an equivalent one using exponential functions, as described in Shtoff (2024b).

The exponential function in the cost induces a difficulty for gradient methods. From a theoretical perspective, the cost and all its derivatives are unbounded, even if the parameters $\boldsymbol{\theta}, \boldsymbol{\phi}, b, a$ are bounded. From a practical perspective, we may encounter floating-point overflows during training due to the exponentiation of possible large numbers. As shown in Asi & Duchi (2019), the model based minimization framework does not suffer from these theoretical issues if the training data that is manifested in the parameters $\boldsymbol{\theta}, \boldsymbol{\phi}, b$ is properly normalized. Moreover, as we shall see in this paper, the framework also does not suffer from the numerical issues of over-flows. Thus, our work facilitates a more reliable training procedure when training with cost functions involving exponentiation.

The work of Shtoff (2024a) develops several frameworks facilitating efficient and easy to implement algorithms for computing proximal operators of many cost function families useful in machine learning, and utilizes the framework to devise algorithms and code for many concrete examples. Our paper can be seen as a direct application of one of the frameworks devised in Shtoff (2024a) to tackle the family described in equation 1.

[1]in this context, a "model" is not a machine-learned model, but rather a family of approximating functions that attempt to model the cost ℓ

The main contributions of our work are:

- a closed-form formula for $\text{prox}_{\eta f}$, where f is of the form in equation 1, in terms of the well-known Lambert-W function;
- a reformulation of the above formula using the Wright-Omega function (Corless & Jeffrey, 2002), available in SciPy, to obtain a more numerically-favorable formula;
- a vectorized implementation of the Wright-Omega function in PyTorch (Ansel et al., 2024), and a vectorized implementation of $\text{prox}_{\eta f}$ in both PyTorch and SciPy.

We note that a proximal operator for the so-called piece-wise exponential penalties $t \to 1 - \exp(-|x|/\rho)$ were studied in Liu et al. (2023; 2024). These works, too, obtained formulae that rely on the Lambert W function. Beyond the apparent similarity because of the use of the exponential function, the piecewise-exponential penalty is fundamentally different from the functions we study in this paper in equation 1. One is non-convex, whereas the other is convex. However, the appearance of the exponential function in both, and of the Lambert W function in their proximal operators, may not be a mere coincidence.

2 Preliminaries

Here, we recall some mathematical background, including the algorithmic framework from Shtoff (2024a). Then, we use the results to devise our proximal operator formula.

In this section we use concepts from convex analysis that are typically presented using the formalism of extended real-valued functions. Since in this paper we apply the preliminaries to convex functions defined on the entire space, we specialize all preliminaries to regular convex functions to make this paper accessible to a wider audience without degrading its correctness.

2.1 Lambert's W function

Euler (Euler, 1783), based on the work of Lambert (Lambert, 1758), studied the solution set for y of the equation

$$y \exp(y) = z$$

over the complex numbers. The solution set is described by a family of so-called Lambert W functions $W_k(z)$ for $k \in \{0, 1, 2, ...\}$. In this paper we focus on the solution set over the reals for $z > 0$, which corresponds to $W_0(z)$. See Corless et al. (1996) for a thorough introduction.

For simplicity, we denote $W_0(z)$ by $W(z)$, and refer to it as "the" Lambert W function. We note that Lambert W function is implemented in SciPy (Virtanen et al., 2020) as `scipy.special.lambertw`, in its full generality, for complex numbers and for any k.

2.2 Moreau envelope

To describe the framework, we need to recall another concept related to the proximal operator - the Moreau Envelope (Moreau, 1962; 1965). For a function r, its Moreau envelope is:

$$M_r(\boldsymbol{w}) = \inf_{\boldsymbol{u}} \left\{ r(\boldsymbol{u}) + \frac{1}{2}\|\boldsymbol{u} - \boldsymbol{w}\|_2^2 \right\}.$$

Beyond the fact that the proximal operator is the minimizer of the same objective function, the Moreau envelope has another tight relation to the proximal operator, as shown in the following Lemma.

Lemma 1 (Theorem 6.55 in Beck (2017)). *Let $r : \mathbb{R}^d \to \mathbb{R}$ be a convex function, and let M_r be its Moreau envelope. Then M_r is continuously differentiable, and its gradient is given by:*

$$\nabla M_r(\boldsymbol{w}) = \boldsymbol{w} - \text{prox}_r(\boldsymbol{w}) \tag{2}$$

2.3 Convex conjugate

The convex conjugate of the function ψ is the function ψ^* defined by

$$\psi^*(\boldsymbol{y}) = \sup_{\boldsymbol{u}} \left\{ \langle \boldsymbol{u}, \boldsymbol{w} \rangle - \psi(\boldsymbol{w}) \right\}.$$

The convex conjugate is a central object in optimization, and has many applications and properties. Moreover, tables of conjugate pairs for many useful functions have been devised in the literature. See Beck (2017) for in depth introduction and a comprehensive table of such functions, and the associated calculus properties. In particular, for the exponential function $\psi(t) = \exp(t)$ we have

$$\psi^*(s) = s \ln(s) - s,$$

defined for $s \geq 0$ with the convention that $0 \ln(0) \equiv 0$.

2.4 Algorithmic framework for regularized convex-on-linear proximal operator

The work of Shtoff (2024a) develops a generic framework for computing the proximal operator of functions of the the so-called *regularized convex-on-linear* form:

$$g(\boldsymbol{w}) = \psi(\langle \boldsymbol{\theta}, \boldsymbol{w} \rangle + b) + r(\boldsymbol{w}),$$

where ψ is a univariate convex function, which typically has the role of a loss, and r is a convex function that typically has the role of a regularizer. All the results below in this section were devised in the above work.

The resulting algortihmic framework for computing the proximal operator

$$\operatorname{prox}_{\eta g}(\boldsymbol{w}) = \arg\min_{\boldsymbol{u}} \left\{ g(\boldsymbol{u}) + \frac{1}{2\eta} \|\boldsymbol{u} - \boldsymbol{w}\|_2^2 \right\}$$

comprises the following three steps:

1. form the univariate function:

$$q(s) = M_{\eta r}(\boldsymbol{w} - \eta s \boldsymbol{\theta}) + (\langle \boldsymbol{\theta}, \boldsymbol{w} \rangle + b)s - \frac{\eta \|\boldsymbol{\theta}\|_2^2}{2} s^2 - \psi^*(s) \tag{3}$$

2. compute the unique maximizer $s^* = \arg\max q(s)$;

3. output $\operatorname{prox}_{\eta g}(\boldsymbol{w}) = \operatorname{prox}_{\eta r}(\boldsymbol{w} - \eta s^* \boldsymbol{\theta})$.

The function $q'(s)$ is always concave. In many cases encountered in practice it is also differentiable, and is maximized at the unique solution of $q'(s) = 0$. When h^* is differentiable, so is q, and its derivative is given by:

$$q'(s) = \langle \boldsymbol{\theta}, \operatorname{prox}_{\eta r}(\boldsymbol{w} - \eta s \boldsymbol{\theta}) \rangle - {\psi^*}'(s) + b. \tag{4}$$

For the derivative, the only computational tools that we require is the proximal operator of r, and the convex conjugate of h.

3 Deriving the closed-form formula

In this section we use the algorithmic framework for the proximal operator of regularized convex-on-linear functions to devise a closed-form formula for the function family in equation 1.

3.1 Using the algorithmic framework

The functions having the form in equation 1 are an instance of the family used by the algorithmic framework in section 2.4. Indeed, letting $\psi = \exp$, we can decompose the function f as:

$$f(\boldsymbol{w}; \boldsymbol{\theta}, \boldsymbol{\phi}, b, \alpha) = \underbrace{\exp(\langle \boldsymbol{\theta}, \boldsymbol{w} \rangle + b)}_{\psi(\langle \boldsymbol{\theta}, \boldsymbol{w} \rangle + b)} + \underbrace{\langle \boldsymbol{\phi}, \boldsymbol{w} \rangle + \frac{\alpha}{2} \|\boldsymbol{w}\|_2^2}_{r(\boldsymbol{w})}. \tag{5}$$

Since we already know the convex conjugate $\psi^*(s) = s \ln(s) - s$, our missing ingredient is the proximal operator $\operatorname{prox}_{\eta r}$, but it's quite easy to derive.

Lemma 2. *The proximal operator* $\operatorname{prox}_{\eta r}(\boldsymbol{w})$ *of the function*

$$r(\boldsymbol{w}) = \langle \boldsymbol{\phi}, \boldsymbol{w} \rangle + \frac{\alpha}{2} \|\boldsymbol{w}\|_2^2$$

is given by

$$\operatorname{prox}_{\eta r}(\boldsymbol{w}) = \frac{\boldsymbol{w} - \eta \boldsymbol{\phi}}{1 + \eta \alpha} \tag{6}$$

It is possible to prove this Lemma using calculus properties of proximal operators, but we believe that a direct proof with elementary calculus is both extremely concise, and much easier to grasp for a wide audience.

Proof. We have:

$$r(\boldsymbol{u}) + \frac{1}{2\eta} \|\boldsymbol{u} - \boldsymbol{w}\|_2^2 = \langle \boldsymbol{\phi}, \boldsymbol{u} \rangle + \frac{\alpha}{2} \|\boldsymbol{u}\|_2^2 + \frac{1}{2\eta} \|\boldsymbol{u} - \boldsymbol{w}\|_2^2$$

The above function is a convex quadratic and is minimized over $\boldsymbol{u}$ when its gradient w.r.t $\boldsymbol{u}$ is equal to zero:

$$\boldsymbol{\phi} + \alpha \boldsymbol{u} + \frac{1}{\eta}(\boldsymbol{u} - \boldsymbol{w}) = \mathbf{0}.$$

Solving for $\boldsymbol{u}$, we obtain that the minimizer is

$$\boldsymbol{u}^* = \frac{\boldsymbol{w} - \eta \boldsymbol{\phi}}{1 + \eta \alpha}.$$

Thus, the proximal operator of ηr is

$$\operatorname{prox}_{\eta r}(\boldsymbol{w}) = \arg\min_{\boldsymbol{u}} \left\{ r(\boldsymbol{u}) + \frac{1}{2\eta} \|\boldsymbol{u} - \boldsymbol{w}\|_2^2 \right\} = \frac{\boldsymbol{w} - \eta \boldsymbol{\phi}}{1 + \eta \alpha}.$$

□

We are now ready to show how to maximize the function q given in equation 3, that corresponds to the decomposition in equation 5.

Lemma 3. *The function $q(s)$ defined in equation 3 corresponding to the regularized convex-on-linear decomposition in equation 5 has a unique maximum $s^* > 0$ given by*

$$\begin{aligned} s^* &= \frac{1}{\gamma} \mathrm{W}(\exp(\delta + \ln(\gamma))), \\ \textit{where: } \gamma &= \frac{\eta \|\boldsymbol{\theta}\|_2^2}{1 + \eta \alpha}, \\ \delta &= \frac{\langle \boldsymbol{\theta}, \boldsymbol{w} - \eta \boldsymbol{\phi} \rangle}{1 + \eta \alpha} + b. \end{aligned} \tag{7}$$

Proof. ψ^* is differentiable, and its derivative is given by ${\psi^*}'(s) = \ln(s)$. Since ψ^* is differentiable, so is $q(s)$, and by equation 4 its derivative is given by

$$\begin{aligned} q'(s) &= \langle \boldsymbol{\theta}, \operatorname{prox}_{\eta r}(\boldsymbol{w} - \eta s \boldsymbol{\theta}) \rangle - {\psi^*}'(s) + b \\ &= \left\langle \boldsymbol{\theta}, \frac{\boldsymbol{w} - \eta s \boldsymbol{\theta} - \eta \boldsymbol{\phi}}{1 + \eta \alpha} \right\rangle - \ln(s) + b \\ &= -\frac{\eta \|\boldsymbol{\theta}\|_2^2}{1 + \eta \alpha} s - \ln(s) + \left\langle \boldsymbol{\theta}, \frac{\boldsymbol{w} - \eta \boldsymbol{\phi}}{1 + \eta \alpha} \right\rangle + b \\ &= -\gamma s - \ln(s) + \delta. \end{aligned}$$

We can see that $q'(s)$ is strictly decreasing on $s > 0$, and thus q is strictly concave. Thus, if the equation $q'(s) = 0$ has a unique solution in $s > 0$, this must be the unique optimum. Adding $\ln(\gamma)$ to both sides of the equation $q'(s) = 0$ and simplifying, we obtain

$$\gamma s + \ln(\gamma s) = \delta + \ln(\gamma).$$

Exponentiating both sides, we get

$$(\gamma s) \exp(\gamma s) = \exp(\delta + \ln(\gamma)).$$

By definition of the Lambert W function, the above is equivalent to

$$\gamma s = \mathrm{W}(\exp(\delta + \ln(\gamma))),$$

and therefore the unique solution is

$$s^* = \frac{1}{\gamma} \mathrm{W}(\exp(\delta + \ln(\gamma))),$$

as required. ☐

We now have a full algorithm for computing the proximal operator $\operatorname{prox}_{\eta f}$ of functions f having the form in equation 1:

1. Compute s^* according to Equation equation 7,
2. Output:

$$\operatorname{prox}_{\eta f}(\boldsymbol{w}) = \frac{\boldsymbol{w} - \eta s^* \boldsymbol{\theta} - \eta \boldsymbol{\phi}}{1 + \eta \alpha}.$$

Although it may appear that we are done, careful inspection of equation 7 shows that computing s^* by definition requires computing $\exp(\delta + \ln(\gamma))$, which may lead to an overflow with floating point arithmetic. However, intuitively we understand that $W(x)$ acts as a kind of a logarithm, since it is the inverse function of $y \exp(y)$, and hence it should "cancel out" the effect of the exponentiation. The next section rigorously deals with this issue and provides a formula for computing s^* that avoids exponentiation altogether.

3.2 Avoiding exponentiation with the Wright-Omega function

The Wright-Omega function $\omega(z)$ (Corless & Jeffrey, 2002) for a *real*[2] argument z is defined by

$$\omega(z) = W(\exp(z)).$$

By definition of the Lambert W function, one can also see that $\omega(z)$ is the solution set of the equation

$$y + \ln(y) = z,$$

[2]Corless & Jeffrey (2002) present the definition for arbitrary complex numbers, but in this paper we specialize their definition to real numbers only

for y. Based on this observation, Lawrence et al. (2012) have devised a fast and direct numerical method for computing $\omega(z)$ *without* relying on the Lambert W function, and it has been implemented in SciPy as in `scipy.special.wrightomega`. Therefore, we can reformulate equation 7 for computing the solution of $q'(s) = 0$ as

$$\begin{aligned} s^* &= \frac{1}{\gamma}\omega(\delta + \ln(\gamma)), \\ \text{where: } \gamma &= \frac{\eta\|\boldsymbol{\theta}\|_2^2}{1+\eta\alpha}, \\ \delta &= \frac{\langle \boldsymbol{\theta}, \boldsymbol{w} - \eta\boldsymbol{\phi}\rangle}{1+\eta\alpha} + b. \end{aligned} \tag{8}$$

3.3 Open source implementation

We have created an open-source implementation of our proximal operator algorithm, both for NumPy arrays and PyTorch tensors. Both implementations support additional mini-batch dimensions prepended to the argument $\boldsymbol{w}$, and the parameters of the function f. Since PyTorch does not include an implementation of the Wright-Omega function, we have also ported the SciPy implementation to PyTorch in our code repository as well. The code can be found at `https://anonymous.4open.science/r/exponential-proximal-point-B8DD`, and includes tests to verify the correctness by comparing the resulting computation to the solutions obtained by CVXPY (Diamond & Boyd, 2016; Agrawal et al., 2018).

4 Summary

In this work we have devised a closed-form expression in terms of the Wright-Omega special function for the proximal operator for a family of functions that appears as regularized losses mainly in Poisson regression, but also in other possible applications. Our work allows researchers working incremental proximal point algorithms to perform numerical experiments with yet another machine-learning application of Poisson regression, and practitioners to use our work whenever their training procedure can potentially cause numerical overflows. We hope that in addition to other well-known “special” functions, such as the Gamma function, the Lambert W and the Wright-Omega function make it into additional machine-learning frameworks as first-class objects.

References

Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, C. K. Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Shunting Zhang, Michael Suo, Phil Tillet, Xu Zhao, Eikan Wang, Keren Zhou, Richard Zou, Xiaodong Wang, Ajit Mathews, William Wen, Gregory Chanan, Peng Wu, and Soumith Chintala. Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. ASPLOS ’24, pp. 929–947, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400703850. doi: 10.1145/3620665.3640366. URL `https://doi.org/10.1145/3620665.3640366`.

Hilal Asi and John C. Duchi. Stochastic (approximate) proximal point methods: Convergence, optimality, and adaptivity. *SIAM Journal on Optimization*, 29(3):2257–2290, 2019. doi: 10.1137/18M1230323. URL `https://doi.org/10.1137/18M1230323`.

Amir Beck. *First-Order Methods in Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2017. doi: 10.1137/1.9781611974997. URL `https://epubs.siam.org/doi/abs/10.1137/1.9781611974997`.

R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth. On the lambertw function. *Advances in Computational Mathematics*, 5(1):329–359, Dec 1996. ISSN 1572-9044. doi: 10.1007/BF02124750. URL https://doi.org/10.1007/BF02124750.

Robert M. Corless and D. J. Jeffrey. The wright ωfunction. In Jacques Calmet, Belaid Benhamou, Olga Caprotti, Laurent Henocque, and Volker Sorge (eds.), *Artificial Intelligence, Automated Reasoning, and Symbolic Computation*, pp. 76–89, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

Damek Davis and Dmitriy Drusvyatskiy. Stochastic model-based minimization of weakly convex functions. *SIAM Journal on Optimization*, 29(1):207–239, 2019. doi: 10.1137/18M1178244. URL https://doi.org/10.1137/18M1178244.

Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011. URL http://jmlr.org/papers/v12/duchi11a.html.

Leonhard Euler. De serie lambertina plurimisque eius insignibus proprietatibus. *Acta Academiae scientiarum imperialis petropolitanae*, pp. 29–51, 1783.

Geoffrey J. Gordon. Regret bounds for prediction problems. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, COLT '99, pp. 29–40, New York, NY, USA, 1999. Association for Computing Machinery. ISBN 1581131674. doi: 10.1145/307400.307410. URL https://doi.org/10.1145/307400.307410.

Johann Heinrich Lambert. Observationes variae in mathesin puram. *Acta Helvetica*, 3(1):128–168, 1758.

Piers W. Lawrence, Robert M. Corless, and David J. Jeffrey. Algorithm 917: Complex double-precision evaluation of the wright ω function. *ACM Trans. Math. Softw.*, 38(3), April 2012. ISSN 0098-3500. doi: 10.1145/2168773.2168779. URL https://doi.org/10.1145/2168773.2168779.

Tian Li, Ahmad Beirami, Maziar Sanjabi, and Virginia Smith. On tilted losses in machine learning: Theory and applications. *Journal of Machine Learning Research*, 24(142):1–79, 2023. URL http://jmlr.org/papers/v24/21-1095.html.

Yulan Liu, Yuyang Zhou, and Rongrong Lin. The proximal operator of the piece-wise exponential function and its application in compressed sensing. *arXiv preprint arXiv:2306.13425*, 2023.

Yulan Liu, Yuyang Zhou, and Rongrong Lin. The proximal operator of the piece-wise exponential function. *IEEE Signal Processing Letters*, 31:894–898, 2024. doi: 10.1109/LSP.2024.3370493.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

H Brendan McMahan and Matthew Streeter. Adaptive bound optimization for online convex optimization. *arXiv preprint arXiv:1002.4908*, 2010.

Jean Jacques Moreau. Fonctions convexes duales et points proximaux dans un espace hilbertien. *Comptes rendus hebdomadaires des séances de l'Académie des sciences*, 255:2897–2899, 1962.

Jean-Jacques Moreau. Proximité et dualité dans un espace hilbertien. *Bulletin de la Société mathématique de France*, 93:273–299, 1965.

J. A. Nelder. Log linear models for contingency tables: A generalization of classical least squares. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 23(3):323–329, 1974. ISSN 00359254, 14679876. URL http://www.jstor.org/stable/2347125.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951. ISSN 00034851. URL http://www.jstor.org/stable/2236626.

Alex Shtoff. Efficient algorithms for implementing incremental proximal-point methods. *Mathematical Programming Computation*, 16(3):423–458, Sep 2024a. ISSN 1867-2957. doi: 10.1007/s12532-024-00258-8. URL https://doi.org/10.1007/s12532-024-00258-8.

Alex Shtoff. Untilting the tilted loss. https://alexshtf.github.io/2024/06/14/Untilting.html, 2024b. Accessed: 2024-11-21.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*, pp. 928–936, 2003.