Discretize mean-field limit using JKO, see if it is similar to GD.

1 Classic setup

Data $x_j \in \mathbb{R}^d$ and labels $y_j \in \mathbb{R}$, $j = 1, .., n$

First layer $w_i \in \mathbb{R}^d$, second layer $\alpha_i \in \mathbb{R}$, $i = 1, .., m$

$\gamma > 0$ step-size, β regularization

$$\mathcal{L}(W, \alpha) = \sum_{j=1}^{n} \Big(\underbrace{\sum_{i=1}^{m} \max(0, w_i^\top x_j)\alpha_i}_{\text{Network's Output}} - y_j \Big)^2 + \underbrace{\lambda \sum_{i=1}^{m} \|w_i\|_2^2 + \alpha_i^2}_{\text{Weight Decay}}$$

Discret time.

Full-batch gradient descent

$$(W, \alpha)_{t+1} = (W, \alpha)_t - \gamma \nabla \mathcal{L}((W, \alpha)_t)$$

Implicit

$$\theta_{t+1} = \arg\min_{\theta} \mathcal{L}(\theta) + \frac{1}{2\gamma} \|\theta - \theta_t\|$$

Continuous time.

Taking $\gamma \to 0$, we get the gradient flow: $\frac{\mathrm{d}\theta_t}{\mathrm{d}t} = -\nabla \mathcal{L}(\theta_t)$. We make ReLU differentiable with $\sigma'(0) = 0$ as justified in (Boursier et al.).

2 Using a measure

Mean-field limit(Chizat & Bach): For a sufficiently large width, the training dynamics of a NN can be coupled with the evolution of a probability distribution described by a PDE.

If [...] converges, with $m \to \infty$ (many-particle limit), our particles of interest converges to a Wasserstein gradient flow of F:

$$\partial \mu_t = -\mathrm{div}(v_t \mu_t) \text{ where } v_t \in -\partial F'(\mu_t)$$

$$\int_{\Theta} m(\theta; x) \mathrm{d}\mu(\theta) = \frac{1}{m} \sum_{i=1}^{m} \langle \boldsymbol{w}_i, \boldsymbol{x}_j \rangle_+ \alpha_i$$

Different ways to use a measure:

- $\Theta = \mathbb{R}^d \times \mathbb{R}$, measure $\mu = \frac{1}{m} \sum_{i=1}^{m} \delta_{\theta_i = (w_i, \alpha_i)}$, output of one neuron $m(\theta = (\boldsymbol{w}, \alpha); \boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{w} \rangle_+ \alpha$: (works, output matches discrete)
- $\Theta = \mathbb{R}^d$, measure $\mu = \frac{1}{m} \sum_{i=1}^{m} \alpha_i \delta_{\theta_i = w_i}$ output of one neuron $m(\theta = \boldsymbol{w}; \boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{w} \rangle_+$ (works)
- $\Theta = \mathbb{R}^d \times \mathbb{R}^d$, output of one neuron $m(\tilde{\boldsymbol{w}}_+, \tilde{\boldsymbol{w}}_-, \boldsymbol{x}) = \langle \tilde{\boldsymbol{w}}_+, \boldsymbol{x} \rangle - \langle \tilde{\boldsymbol{w}}_-, \boldsymbol{x} \rangle$ (works, separate neg and positive)
- $\Theta = (S^{d-1} \times \mathbb{R})$, output of one neuron $m((\boldsymbol{d}, \tilde{\alpha}); \boldsymbol{x}) = \tilde{\alpha} \langle \boldsymbol{d}, \boldsymbol{x} \rangle = \tilde{\alpha}\ \mathbb{1}_{\langle \boldsymbol{d}, \boldsymbol{x} \rangle > 0}$ (works), mapping: $\boldsymbol{d} = \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$ and $\tilde{\alpha} = \|\boldsymbol{w}\| \alpha$. Gradient are not equal to discrete.

2.1 Algorithm, discretize the measure's space

Take a grid of N points in Θ, we can match the notation above by taking a neuron for each point of the grid $m = N$.

$$\mu(t+1) = \underset{\mu \in \mathcal{M}(\Theta)}{\arg\min} F(\mu) + \frac{1}{2\gamma} W_2(\mu; \mu(t))$$

A essayer: KL à la place de distance wasserstein.

Remark: le 1/m c'est principalement pour être ok à l'infini. Dans le papier JKO (Carlier et al.) ils utilisent un vecteur de proba

2.2 Infinity and beyond

Take $\gamma \to 0$, get gradient flow. Take $m \to \infty$, get wasserstein gradient flow (Bach & Chizat), and if it converges, it goes to the global optimal.

2.3 JKO

What we compute by using the entropic JKO flow iterations.

$$\begin{aligned}
\forall t > 0, p_{t+1} &:= \mathrm{Prox}_{\tau f}^{W_\gamma}(p_t) \\
&= \underset{p \in \text{simplex}}{\arg\min}\ W_\gamma(p, q) + \tau f(p) \\
&= \underset{p \in \text{simplex}}{\arg\min} \left(\min_{\pi \in \Pi(p,q)} \langle c, \pi \rangle + \gamma E(\pi) \right) + \tau f(p)
\end{aligned}$$

f "should" be convex and with a closed form proximal

- Meta Optimal Transport (paper) and (code git): InputConvexNN to predict solution of OT problem
- JKOnet (paper) and (code git):
  - /models -> sinkhorn loss defined in loss.py, differentiable loop in fixed point.py
  - next step: trying to create the right Geometry object from OTT library, which is what's used for sinkhorn

2.4 Papers

The algo we try to implement

Paper with a specific case that doesn't match ours:

In the future, large-scale waserstein gradient flows

2.4.1 Grid problems

The grid currently dictate the neuron's scale, giving multiple choices. One solution: duplicate each neuron, make one with a small scale and one with a very big scale.

References

Francis Bach and Lenaïc Chizat. Gradient descent on infinitely wide neural networks: Global convergence and generalization. URL http://arxiv.org/abs/2110.08084.

Etienne Boursier, Loucas Pillaud-Vivien, and Nicolas Flammarion. Gradient flow dynamics of shallow relu networks for square loss and orthogonal inputs. URL http://arxiv.org/abs/2206.00939.

Guillaume Carlier, Vincent Duval, Gabriel Peyré, and Bernhard Schmitzer. Convergence of entropic schemes for optimal transport and gradient flows. URL http://arxiv.org/abs/1512.02783.

Lénaïc Chizat and Francis R. Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Adv. Neural Inf. Process. Syst. 31 Annu. Conf. Neural Inf. Process. Syst. 2018 NeurIPS 2018 Dec. 3-8 2018 Montr. Can.*, pp. 3040–3050. URL https://proceedings.neurips.cc/paper/2018/hash/a1afc58c6ca9540d057299ec3016d726-Abstract.html.