Supervised Learning with Black Box Loss Functions

David S. Rosenberg

NYU: CDS

November 3, 2021

Contents

Black box and non-differentiable losses

Supervised learning

- Input space $\mathcal{X}$; Label space $\mathcal{Y}$
- Hypothesis space of functions $x \mapsto f_\theta(x) \in \mathcal{Y}$
- For $(X, Y) \sim P$, loss function $\ell(f_\theta(X), Y)$ tells us how we did.
- For gradient-based learning methods, we compute

$$\nabla_\theta \ell(f_\theta(X), Y) = \frac{\partial \ell}{\partial f_\theta(X)} \ell(f_\theta(X), Y) \nabla_\theta f_\theta(X)$$

and iteratively update θ as $\theta \leftarrow \theta - \eta \nabla_\theta \ell(f_\theta(X), Y)$

What about discontinuous or black-box losses?

- For gradient-based learning methods, we compute

$$\nabla_\theta \ell(f_\theta(X), Y) = \frac{\partial \ell}{\partial f_\theta(X)} \ell(f_\theta(X), Y) \nabla_\theta f_\theta(X)$$

- Example 1: loss not differentiable w.r.t. action
  - e.g. Linear classifier, 0/1 loss: $f_\theta(x) = \theta^T x$ and $\ell(a, y) = \mathbb{1}[ay < 0]$. $(y \in \{-1, 1\})$
  - $\ell(a, y)$ not differentiable w.r.t. action a (not even continuous).
- Example 2: discrete action space
  - e.g. Hard classifier $f_\theta(x) = \arg\max_{a \in \mathcal{A}} \pi_\theta(a \mid x)$ with any $\ell(a, y)$
  - Not even clear what $\frac{\partial}{\partial a}\ell(a, y)$ means with a discrete action space $\mathcal{A}$.
  - $f_\theta(x)$ not differentiable (not even continuous) w.r.t. θ, for discrete $\mathcal{A}$.
- In each scenario, $\theta \mapsto \ell(f_\theta(X), Y)$ is piecewise constant, so gradient has no useful information.

Examples

- 0/1 loss for classification
    - discontinuous loss function
- BLEU score – used in machine translation for scoring a candidate sentence against a reference set of translations
    - applied to discrete action space (sentences)
- ROUGE metrics – used in automatic summarization for scoring summaries against a reference set of summaries
    - applied to discrete action spaces (sentences/paragraphs)
- Black box loss function
    - if we don't know the loss function, we also don't know its gradient

The essence of the issue

- Gradient methods are iterative and "local".
- Suppose our current model is f_θ.
- By examining $\ell(f_\theta(X), Y)$ in a small neighborhood of θ,
    - we can figure out how to make a small change to θ to improve performance.
    - This is exactly what we learn from the gradient!
- With non-differentiable and discontinuous loss functions, we cannot use the gradient.
- With discrete label spaces:
    - $\theta \mapsto \ell(f_\theta(X), Y)$ will be piecewise constant, and so.
    - we get no information about how to change θ by looking in a small neighborhood of θ.

Randomization and policy gradient

Randomized actions

- Let's relax our setting from deterministic predictions $f_\theta(x) \in \mathcal{Y}$
  - to randomized actions $A \in \mathcal{Y}$ drawn from $\pi_\theta(a \mid x)$.
- Let's consider the expected loss as our performance measure:

$$\mathbb{E}_{A \sim \pi_\theta(a|X)} \ell(A, Y).$$

- Note that deterministic predictions are a special case of randomized predictions.
- We want a conditional probability model on actions that gets small expected loss.
- How has this helped?

Randomness smooths our objective

- Expected loss is

$$\mathbb{E}_{A\sim\pi_\theta(a|X)}\ell(A,Y) = \sum_{a\in\mathcal{A}}\pi_\theta(a\mid X)\ell(a,Y).$$

- Before, we were evaluating the loss $\ell(f_\theta(X),Y)$, where
  - $f_\theta(X)$ may change discontinuously as a function of θ, or
  - $\ell(a,y)$ may not be a smooth function of a.
  - There's no gradient to tell us how to move θ to improve objective.
- For expected loss, we're always evaluating the loss on all possible actions.
- As θ varies, we're changing the relative weights on losses from each action.
- Expected loss changes smoothly as θ changes (so long as $\pi_\theta(a\mid X)$ changes smoothly).

Gradient of expected loss

- Gradient of expected loss:

$$
\begin{aligned}
\nabla_\theta \left[\mathbb{E}_{A\sim\pi_\theta(y|X)}\ell(A,Y)\right] &= \nabla_\theta \left[\sum_{a\in\mathcal{A}} \pi_\theta(a\mid X)\ell(a,Y)\right] \\
&= \sum_{a\in\mathcal{A}} \ell(a,Y)\,\nabla_\theta \pi_\theta(a\mid X)
\end{aligned}
$$

- We can compute the gradient of expected loss so long as $\nabla_\theta \pi_\theta(a\mid X)$ exists.
- We don't need to differentiate w.r.t. the loss.
- What if the action space $\mathcal{A}$ is too large to sum over? (e.g. all English sentences)

Clever trick again

- We have

$$\begin{aligned}\nabla_\theta\left[\mathbb{E}_{A\sim\pi_\theta(a|X)}\ell(A,Y)\right] &= \sum_{a\in\mathcal{A}}\ell(a,Y)\,\nabla_\theta\pi_\theta(a\mid X)\\ &= \sum_{a\in\mathcal{A}}\ell(a,Y)\,\pi_\theta(a\mid X)\nabla_\theta\log\pi_\theta(a\mid X)\\ &= \mathbb{E}_{A\sim\pi_\theta(a|X)}\ell(A,Y)\,\nabla_\theta\log\pi_\theta(A\mid X)\end{aligned}$$

- Now we can use Monte Carlo to estimate this.
- If we have a sample $(X,Y)\sim P$ and $A\sim\pi_\theta(a\mid X)$, then

$$\ell(A,Y)\,\nabla_\theta\log\pi_\theta(A\mid X)$$

is an unbiased estimate of $\nabla_\theta\left[\mathbb{E}_{A\sim\pi_\theta(a|X)}\ell(A,Y)\right]$.

Better gradient estimates with more compute

- In the supervised setting,
  - we know the loss function $\ell(\hat{y}, y)$,
  - so we can get the losses for many possible actions for any given (X, Y).
- If we generate a larger sample $A_1, \ldots, A_n$ of actions from $\pi_\theta(a \mid X)$, then

$$\frac{1}{n}\sum_{i=1}^{n} \ell(A_i, Y)\,\nabla_\theta \log \pi_\theta(A_i \mid X)$$

will be an improved (lower-variance) unbiased estimate of $\nabla_\theta\left[\mathbb{E}_{A\sim\pi_\theta(a|X)}\ell(A, Y)\right]$.

Comparison to policy gradient for contextual bandits

- Gradient estimate for supervised setting for example (X_t, Y_t):

$$\frac{1}{n}\sum_{i=1}^{n} \ell(A_i, Y_t)\,\nabla_\theta \log \pi_{\theta_t}(A_i \mid X_t)$$

- Analogous estimate for policy gradient in contextual bandit setting:

$$R_t(A_t)\nabla_\theta \log \pi_{\theta_t}(A_t \mid X_t)$$

- By averaging over many actions, we get a better estimate of $\nabla_\theta \left[\mathbb{E}_{A\sim\pi_{\theta_t}(a\mid X_t)}\ell(A, Y_t)\right]$.
- Despite the difference, we would still call both of these **policy gradient** methods.

Comparison to maximum likelihood

- We're using conditional probability models.
- And we observe ground truth labels.
- Why not just fit the CPMs with maximum likelihood?
- That's typical, but it ignores our loss function!
- MLE focuses on putting as much weight as possible on the ground truth labels.
- But perhaps the loss function is indifferent among multiple labels.
    - e.g. There are multiple perfect translations, but our "label" is just one of them.
    - Maximizing w.r.t. a loss function that's indifferent to these variations could be better.
- On the negative side, policy gradient methods tend to be much slower than MLE.

Policy gradient for sequence prediction

Application: sequence-to-sequence Models

- Consider machine translation.
- e.g. Conditioned on sentence in English, produce a distribution on sentences in French.
- Model is $\pi_\theta(a \mid x)$, where x is an English sentence and a is a French sentence.
- Typically trained on ground truth pairs (X_i, Y_i) using maximum likelihood:

$$\theta^* = \arg\max_\theta \prod_{i=1}^{n} \pi_\theta(Y_i \mid X_i).$$

- Seems reasonable...
- But how do we actually measure performance for machine translation?

Application: sequence-to-sequence models

- Suppose we are assessing performance of our MT model on a test set.
- We get input X.
- We use our current model $\pi_\theta(\cdot \mid X)$ and produce a sequence A.
  - (e.g. by sampling, Viterbi decoding, beam search, etc.)
- Suppose A is a perfect translation of X, but it's different from the ground truth Y.
- We'd like to give credit for this translation.
- I don't think there's currently a great way to do this without a human in the loop...

But there is BLEU score

- A frequent measure of translation quality is BLEU score.
- Let's not discuss the details of BLEU score.
- For our purposes, it's sufficient to know that
  - BLEU takes a proposed translation and a ground truth and gives a numerical score
- We can use the technique described above to optimize for BLEU score.
- Even though BLEU score is computed by an algorithm and is **not differentiable.**

Exposure bias

- Sequence models are frequently **autoregressive**.
- We condition on previously predicted tokens to predict the next token in a sequence.
- During MLE training, we're always conditioning on the gold labels in Y.
- During test, we're conditioning on our own predicted labels.
- **Our model never trains using its own predictions as input.**

Exposure bias

- This is a known issue with maximum likelihood training of sequence models.
- There is a family of approaches called "learning to search" that address this issue.
- e.g. SEARN, DAgger, AggreVaTe, LOLS, etc.
- Policy gradient addresses this as well.
- Sequences are generated by sampling from $\pi_\theta(\cdot \mid X)$ during training and test.
    - Doesn't involve the ground truth sequence Y at all.

Usually we pre-train with maximum likelihood

- Suppose we want to train a sequence-to-sequence model
  - with BLEU score as reward.
- Policy gradient is sufficient for this task.
- In practice, we usually pre-train our model with maximum likelihood.
  - It's much faster than policy gradient.
- Then switch to policy gradient to optimize to our particular loss/reward.

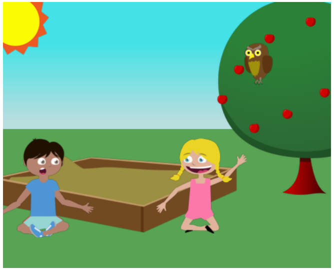
Self-critical baseline[1]

- When we have access
    - to the loss function $\ell(a, y)$ and
    - the ground truth labels $Y_1, Y_2, \ldots,$
- there's a clever way to set a baseline:
    - Find (or approximate) the action that is optimal under our policy: $A_t^* \approx \arg\max_a \pi_{\theta_t}(a|X_t)$
        - using whatever we would usually use to predict a sequence, e.g. beam search
    - Use the loss $\ell(A_t^*, Y_t)$ as a baseline.
- If the current action performs better than the action our policy says is best, then we should make the current action more likely.
- If it performs worse than what the policy says is best, let's make it less likely.

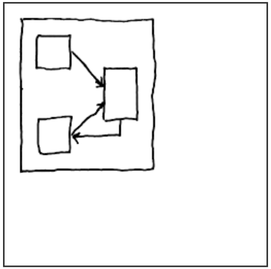
[1] This approach comes from [RMM+17].

Image to Sequence

Image to sequence problems[2]



```
<object>
  <supercategory>C-1</supercategory>
  <category>CS-3<\category>
  <x-coordinate>120</x-coordinate>
  <y-coordinate>240</y-coordinate>
  <depth>1</depth>
  <flip>0</flip>
</object>
<object>....
```



```
<object>
  <category>Rectangle<\category>
  <x1-coordinate>7</x1-coordinate>
  <y1-coordinate>1</y1-coordinate>
  <x2-coordinate>11</x2-coordinate>
  <y2-coordinate>16</y2-coordinate>
</object>
<object>....
```

[2]Results and images for this section are taken from [PRMB21]

- In these image to sequence tasks, there is a domain-specific language (DSL) for specifying images. For the example on the left from the "Abstract Scene" dataset, the image is rendered directly from the specification. For the example on the right from the "Noisy Shapes" dataset, the image is rendered from the specification, and then a person redrew the image, which accounts for the waviness of the lines. The objective is to take the image as input and produce the appropriate DSL specification. See references in [PRMB21] for the sources of these two examples.

How to evaluate?

- One obvious idea is to re-render and check for an exact match.
- This is a very challenging metric.
- We only get positive feedback when we get the image exactly correct.
- Will take a **very long time** to learn this way (at least starting from scratch).
- Doesn't work for hand-drawn shapes

Evaluation metrics

- Two specifications can be very different, yet render to very similar things.
    - e.g. by reordering objects
- Two images may look very different (e.g. at the pixel level), but have similar specifications
    - e.g. by changing a color
- We can evaluate performance in **image space** and in **specification space**.

Image space measure

- We can measure performance in image space with

$$d_{img} = \|I - \Psi(I^R)\|_2^2,$$

where I is the original image vector and I^R is the rendering of the predicted image.

- For the noisy shapes dataset, Ψ is a Gaussian blurring function.
- For the abstract scene dataset, Ψ is identity function.
- Why isn't this differentiable?
- Computing I^R uses a graphics renderer...
- (There are differentiable renderers now... but that's another story.)

Specification space measure: IOU reward

- Our specifications break down into "objects".
- We can look for exact matches between prediction and ground truth at the object level.
- For numeric attributes, we divide range into 20 bins of equal size
  - consider it a match if the bin is correct
- Can summarize matches with precision, recall, F1, etc.
- A common summary in this scenario is **intersection-over-union** (IOU)....

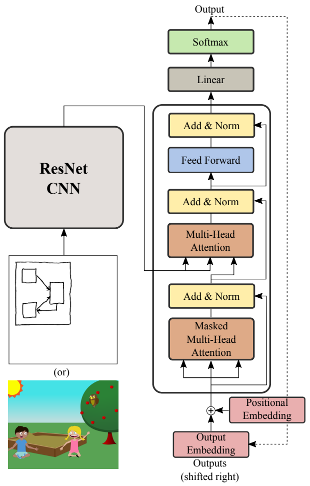
Intersection over union

- Let $\{o_i\}_{i=1}^{m}$ and $\{o_j^*\}_{j=1}^{n}$ represent the objects in predicted and ground-truth specifications, respectively.
- Then the IOU reward is defined as follows:

$$r_{iou} = \frac{\mathrm{count}(\{o_i\}_{i=1}^{m} \cap \{o_j^*\}_{j=1}^{n})}{\mathrm{count}(\{o_i\}_{i=1}^{m} \cup \{o_j^*\}_{j=1}^{n})}$$

- Roughly speaking, IOU gives credit for predicting objects that exactly match objects in the ground truth
- Penalizes both for predicting objects that do not match ground truth objects and for failing to predict objects that are part of the ground truth.

Model: ResNet to transformer decoder



Results: cross-entropy loss (i.e. maximum likelihood)

Model	Recons. Error	IOU
Cross-Entropy Loss		
Image2LSTM+atten.	15.70	32.06
Image2Transformer	10.92	58.54

- reconstruction error corresponds to the image distance
- average error across test set

Results: policy gradient

Model	Recons. Error	IOU
Cross-Entropy Loss		
Image2LSTM+atten.	15.70	32.06
Image2Transformer	10.92	58.54
Image2Transformer with Reinforce Loss		
IOU Reward	10.50	61.29
Recons. Reward	**9.99**	62.44
IOU + Recons.	10.04	**62.45**

References

Resources

- The idea of the self-critical baseline comes from [RMM+17].

References I

[PRMB21] Ramakanth Pasunuru, David S. Rosenberg, Gideon Mann, and Mohit Bansal, *Dual reinforcement-based specification generation for image de-rendering*, Proceedings of the Workshop on Scientific Document Understanding co-located with 35th AAAI Conference on Artificial Inteligence, SDU@AAAI 2021, Virtual Event, February 9, 2021 (Amir Pouran Ben Veyseh, Franck Dernoncourt, Thien Huu Nguyen, Walter Chang, and Leo Anthony Celi, eds.), CEUR Workshop Proceedings, vol. 2831, CEUR-WS.org, 2021.

[RMM$^+$17] Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel, *Self-critical sequence training for image captioning*, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 7 2017, p. nil.