DON'T DECAY THE LEARNING RATE, INCREASE THE BATCH SIZE

SHENGZHE XU

Background

- Stochastic gradient descent
  - Dominant optimization algorithm of deep learning
  - While SDG finds minima that generalize well
    - Each para update only takes a small step

Background

- Previous work declares
  - An optimum fluctuation scale g can be found
    - To maximize the test set accuracy (at constant learning rate)
    - Introduce an optimal batch size that B << N
      - g = epsilon(N/B -1); B, batch size; N, training set size

$$\frac{d\omega}{dt} = -\frac{dC}{d\omega} + \eta(t) \tag{3}$$

the mean $\langle \eta(t) \rangle = 0$ and variance $\langle \eta(t)\eta(t') \rangle = gF(\omega)\delta(t-t')$,

Background

- Large batch training
  - Approach: increase the step size & reduce the number of paras updates required
  - Ad: can be parallelized well and reduce training time
  - Problem: when batch size increase
    - The test set accuracy often falls

Jonathan Hseu, Google Brain Scientist

This Paper

[1709.05011] ImageNet Training in Minutes - arXiv
https://arxiv.org › cs ▾
by Y You - 2017 - Cited by 3 - Related articles
Sep 14, 2017 - Abstract: Finishing 90-epoch **ImageNet**-1k **training** with **ResNet-50** on a NVIDIA M40 GPU takes 14 days. This **training** requires 10^18 single precision operations in total. On the other hand, the world's current fastest supercomputer can finish 2 * 10^17 single precision operations per second (Dongarra et al ...

- This Paper: to train ResNet-50 on ImageNet to 76.1% accuracy in 30 mins.

This Paper

- Existed problem: when one decays the learning rate, one simultaneously decays the g
- Approach: instead of decaying the learning rate,
  - Do increase the batch size during training
  - When lr drops by alpha, increase the batch size by alpha
- Ad:
  - achieves near-identical model performance with same number of training epochs
  - Significantly fewer para updates
  - Doesn't require any fine-tuning

This Paper – further more

- Based on the batch size approach,
  - Able to further reduce the number of para updates
  - By: increase the lr, scaling B pro to epsilon
  - Or increase the momentum coefficient and scale B pro to 1/(1-m).
- Finally:
  - Train Inception-ResNet-V2 on ImageNet under 2500 para updates; using batches of 65536 images, 77% accuracy.
  - Train ResNet-50 on ImageNet to 76.1% accuracy in 30 mins.

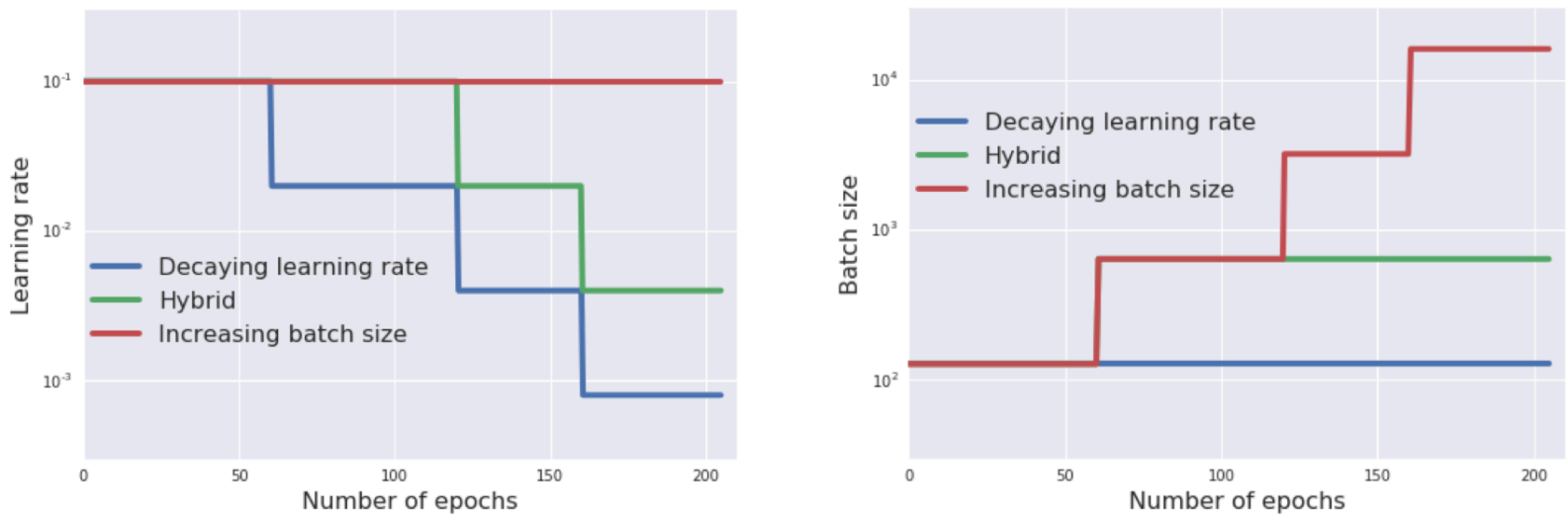
Contribution

- 1. show the quantitatively equivalence of increasing the batch size and decaying the learning rate.
- 2. provide a straightforward pathway towards large batch training.

Details

- Following, we'll shortly see relative thoughts in SDG, decaying learning rates and difficulties of training with large momentum coefficients.
- And then, come back to the experimental evidence.

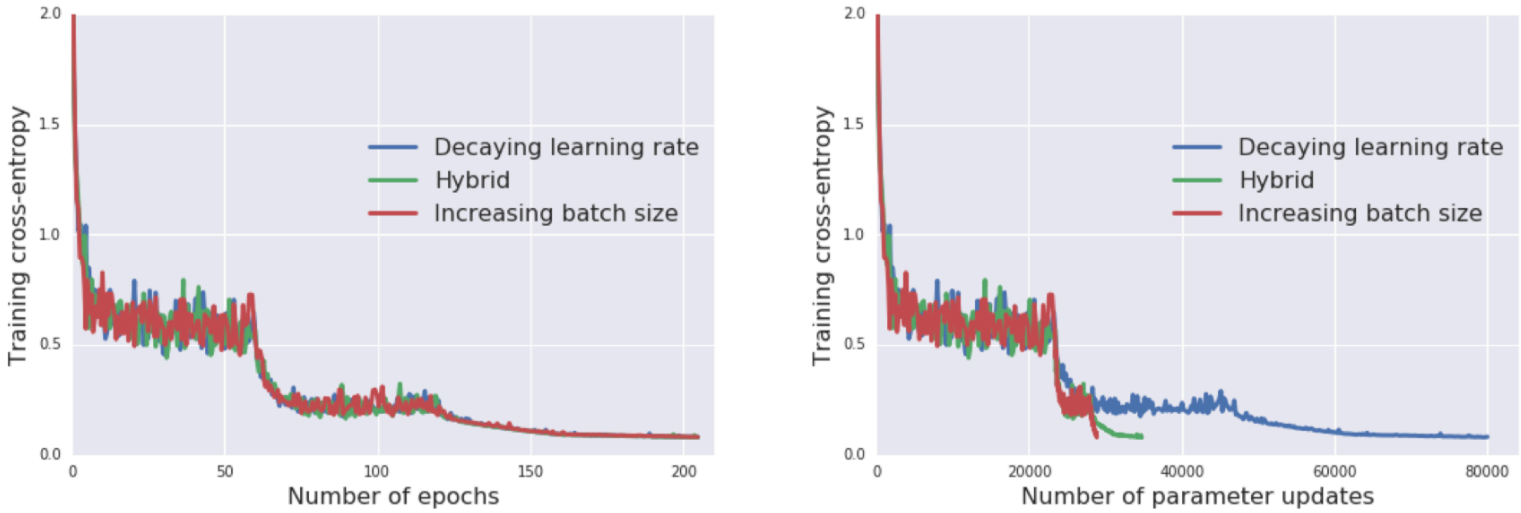
Section 5



(a) (b)

Figure 1: Schedules for the learning rate (a) and batch size (b), as a function of training epochs.

Section 5



(a) (b)

Figure 2: Wide ResNet on CIFAR10. Training set cross-entropy, evaluated as a function of the number of training epochs (a), or the number of parameter updates (b). The three learning curves are identical, but increasing the batch size reduces the number of parameter updates required.

Cross-entropy-1

stackoverflow Questions Developer Jobs Tags Users Search…

58

Cross-entropy is commonly used to quantify the difference between two probability distributions. Usually the "true" distribution (the one that your machine learning algorithm is trying to match) is expressed in terms of a one-hot distribution.

For example, suppose for a specific training instance, the label is B (out of the possible labels A, B, and C). The one-hot distribution for this training instance is therefore:

```
Pr(Class A)  Pr(Class B)  Pr(Class C)
        0.0          1.0          0.0
```

You can interpret the above "true" distribution to mean that the training instance has 0% probability of being class A, 100% probability of being class B, and 0% probability of being class C.

Now, suppose your machine learning algorithm predicts the following probability distribution:

```
Pr(Class A)  Pr(Class B)  Pr(Class C)
      0.228        0.619        0.153
```

How close is the predicted distribution to the true distribution? That is what the cross-entropy loss determines. Use this formula:

$$H(p,q) = -\sum_x p(x) \log q(x).$$

Where `p(x)` is the wanted probability, and `q(x)` the actual probability. The sum is over the three classes A, B, and C. In this case the loss is **0.479** :

```
H = - (0.0*ln(0.228) + 1.0*ln(0.619) + 0.0*ln(0.153)) = 0.479
```

So that is how "wrong" or "far away" your prediction is from the true distribution.

Cross entropy is one out of many possible loss functions (another popular one is SVM hinge loss).

Cross-entropy-2

So to answer your original questions directly:

> Is it only a method to describe the loss function?

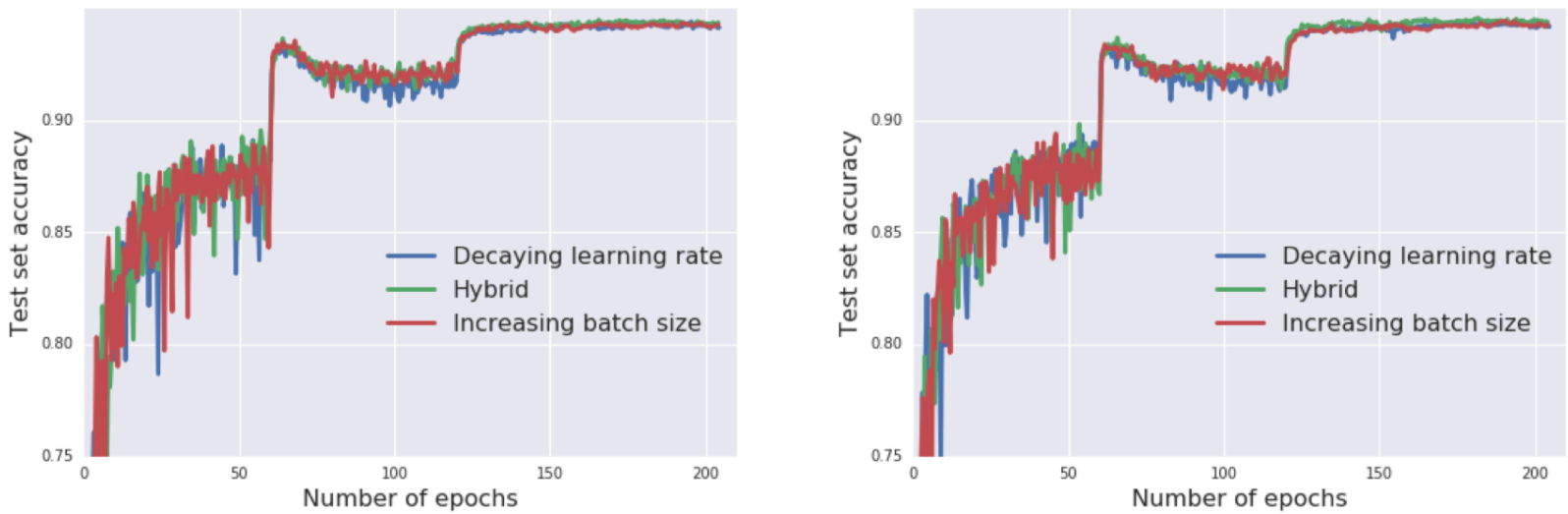
Correct, cross-entropy describes the loss between two probability distributions. It is one of many possible loss functions.

> Then we can use, for example, gradient descent algorithm to find the minimum.

Yes, the cross-entropy loss function can be used as part of gradient descent.

Further reading: one of my other answers related to TensorFlow.

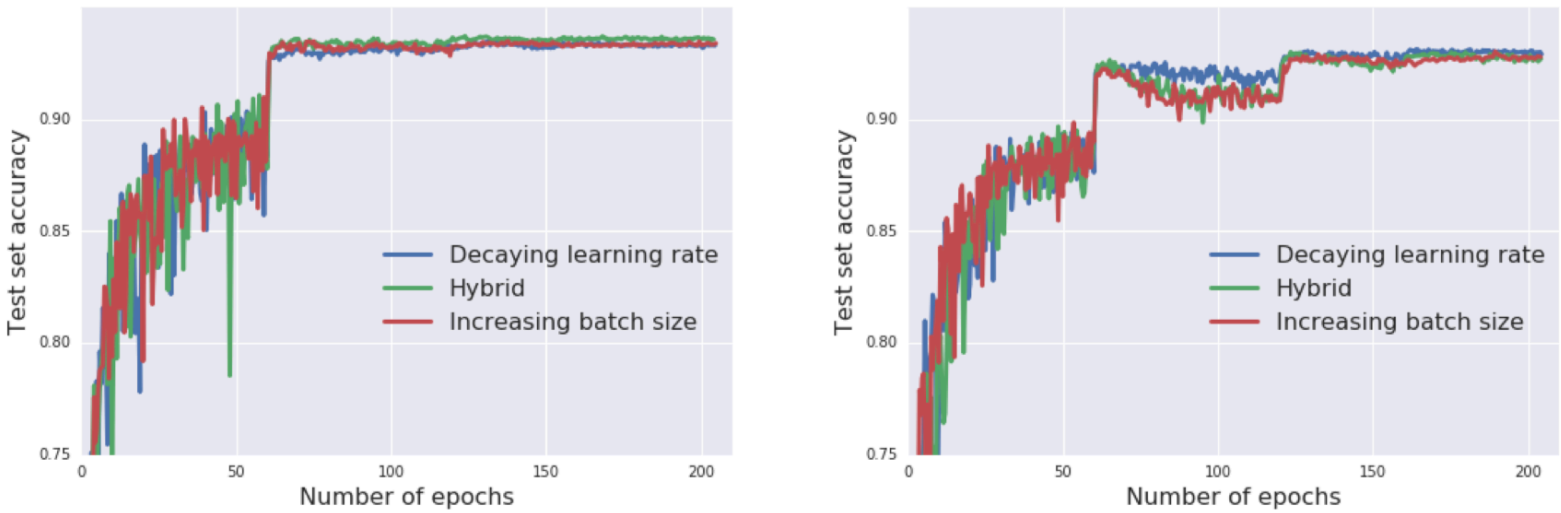
Section 5



(a) (b)

Figure 3: Wide ResNet on CIFAR10. Test accuracy during training, for SGD with momentum (a), and Nesterov momentum (b). In both cases, all three schedules track each other extremely closely.

Section 5



(a) (b)

Figure 4: Wide ResNet on CIFAR10. The test set accuracy during training, for vanilla SGD (a) and Adam (b). Once again, all three schedules result in equivalent test set performance.

Section 5

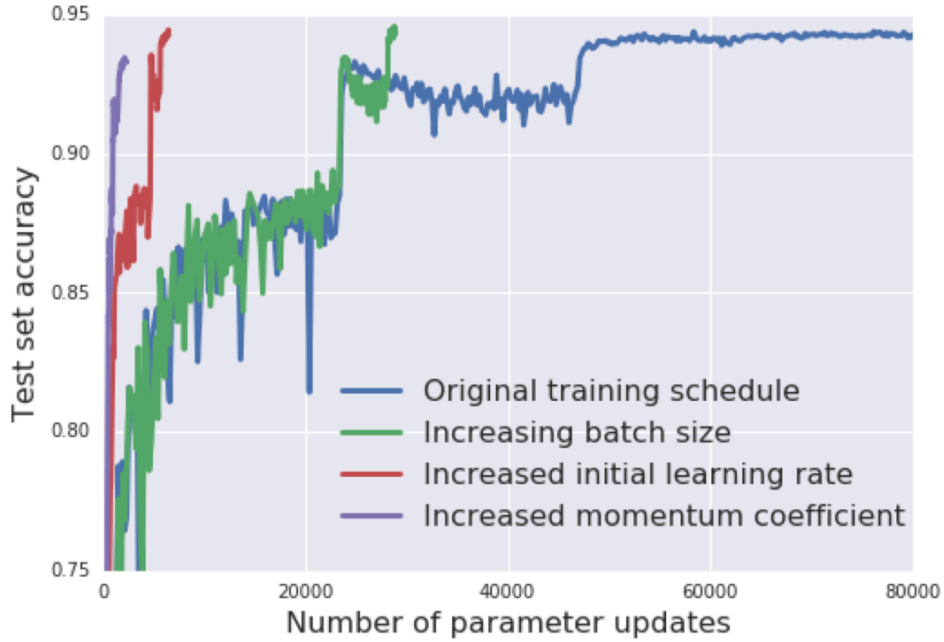


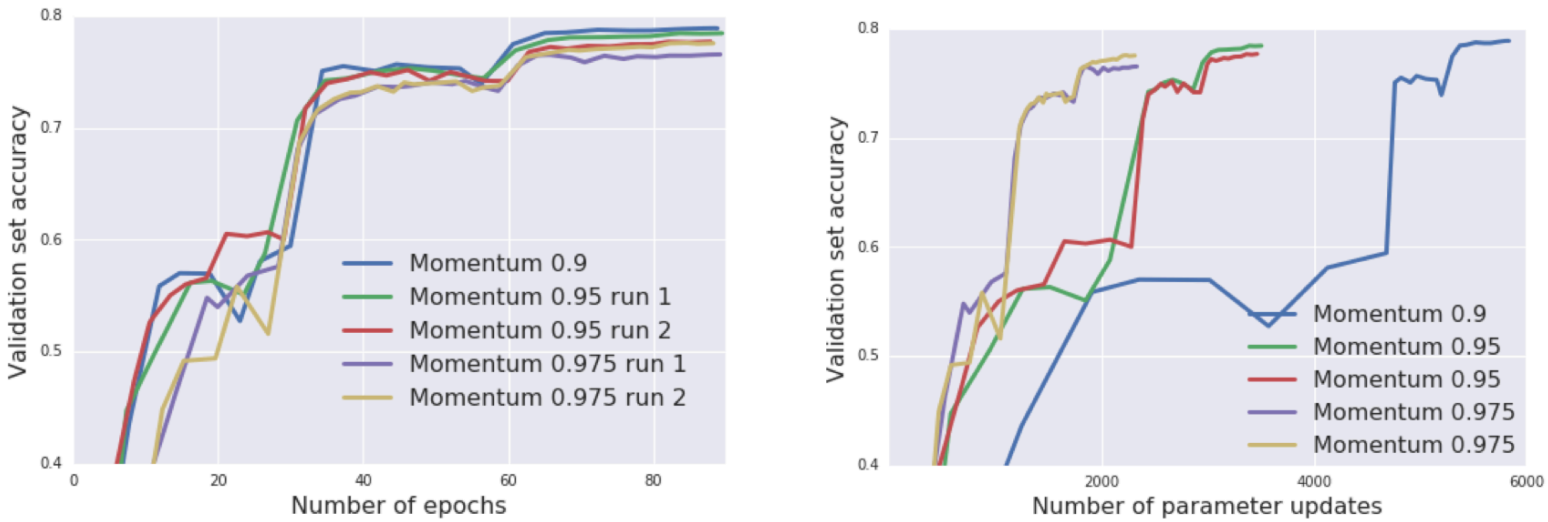
Figure 5: Wide ResNet on CIFAR10. Test accuracy as a function of the number of parameter updates. "Increasing batch size" replaces learning rate decay by batch size increases. "Increased initial learning rate" additionally increases the initial learning rate from 0.1 to 0.5. Finally "Increased momentum coefficient" also increases the momentum coefficient from 0.9 to 0.98.

Section 5

(a) (b)

Figure 6: Inception-ResNet-V2 on ImageNet. Increasing the batch size during training achieves similar results to decaying the learning rate, but it reduces the number of parameter updates from just over 14000 to below 6000. We run each experiment twice to illustrate the variance.

Section 5



(a) (b)

Figure 7: Inception-ResNet-V2 on ImageNet. Increasing the momentum parameter reduces the number of parameter updates required, but it also leads to a small drop in final test accuracy.

Questions?