Deep Learning Recipes and Experiments

Sam Friedman

2/20/2018

Deep Learning Recipes for DNA reads and short variants.

Setting up your environment

We recommend using anaconda to handle your python environments. For CPU only libraries:

```
conda env create -n gatk -f ./gatkcondaenv_cpu.yml
```

To use GPU, you will need a NVIDIA GPU, CUDA and CuDNN installed tensorflow has nice instructions:

```
conda env create -n gatk -f ./gatkcondaenv_gpu.yml
```

Training models from example tensors

In the data directory we provide a small dataset of reference and read tensors from the NA12878 sample. The reference tensors are input for a 1D CNN. They are a 1-hot encoding of 128 base pairs of reference sequence centered at a variant. The read tensors are input for a 2D CNN. They encode reference and read sequence as well as read meta data. They use the tensorflow default channel ordering: reads x sequence x channels. You can toggle between tensorflow and theano channel ordering with the `--channels_last` and `--channels_first` arguments. Uncompress them with tar:

```
cd data
tar -zcvf example_reference_tensors_chr1.tar.gz
tar -zcvf example_read_tensors_chr1_channels_last.tar.gz
cd ..
```

Train a model that predicts variant quality from read tensors and variant annotations:

```
python recipes.py train_ref_read_anno \
  --data_dir ./data/g94982_tensors_chr1_channels_last/ \
  --tensor_map read_tensor \
  --id ref_read_anno_model
```

Train a model that predicts variant quality from read tensors:

```
python recipes.py train_ref_read \
  --data_dir ./data/g94982_tensors_chr1_channels_last/ \
  --tensor_map read_tensor \
  --id ref_read_model
```

Train a model that predicts variant quality from reference sequence and annotations:

```
python recipes.py train_reference_annotation \
  --data_dir ./data/example_reference_tensors_chr1/ \
  --tensor_map reference \
  --id ref_anno_model
```

Train a model that predicts variant quality from reference sequence only:

```
python recipes.py train_reference_annotation \
  --data_dir ./data/example_reference_tensors_chr1/ \
  --tensor_map reference \
  --id ref_model
```

Write tensors with your own data

Create read tensors with a truth vcf, confident region, unfiltered variant calls, and aligned reads:

```
python recipes.py write_tensors \
  --reference_fasta reference.fasta \
  --train_vcf validated_calls.vcf.gz \
  --negative_vcf my_unfiltered_calls.vcf.gz \
  --bed_file validated_calls_confident_region.bed \
  --data_dir ./data/my_read_tensors/ \
  --bam_file my_aligned_reads.bam \
  --tensor_map read_tensor \
  --channels_last \
  --read_limit 128 \
  --window_size 128
```

Create reference tensors with a truth vcf, confident region, and unfiltered variant calls:

```
python recipes.py write_dna_tensors \
  --reference_fasta reference.fasta
  --train_vcf validated_calls.vcf.gz \
  --negative_vcf my_unfiltered_calls.vcf.gz \
  --bed_file validated_calls_confident_region.bed \
  --data_dir ./data/my_reference_tensors/ \
  --tensor_map reference \
  --window_size 128
```

You can downsample specific classes with the `--downsample_class_label` arguments. For example, to only write 10% of the positive SNPs add `--downsample_snps 0.1` to your command line or to keep half of the negative indel examples use: `--downsample_not_indels 0.5`

You can also parallelize over the genome via the `--chrom`, `--start_pos`, and `--end_pos` arguments.