

bpreveal-ga-advanced

June 21, 2023

In this notebook, I'm going to demonstrate a few fitness function concepts, and I'll also show PISA plots of the sequences that the GA invents.

1 Boring stuff

```
[2]: import sys
sys.path.append("/n/projects/cm2363/bpreveal/src/")
import os
os.environ["TF_CPP_MIN_LOG_LEVEL"] = "1"
import tqdm
import utils
utils.setMemoryGrowth()
import gaOptimize
import numpy as np
import pysam
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = [10,8]
import json
import h5py
import scipy

OUTPUT_LENGTH=1000
OUTPUT_START=430700
OUTPUT_END=OUTPUT_START+OUTPUT_LENGTH
INPUT_LENGTH=3092 # This is the input length of my model.
BUFFER=(INPUT_LENGTH - OUTPUT_LENGTH)//2
INPUT_START=OUTPUT_START - BUFFER
INPUT_END=INPUT_START+INPUT_LENGTH
#Offset is how much I shift the mmase endpoints so that
#they line up at where the dyad would be.
OFFSET = 70
# I've copied the model from the pisa strip bias correction work into this
# directory, I'll use it to make predictions.
MODEL_FNAME="models/joint_residual_subtract.model"
GENOME_FNAME="/n/data1/genomes/S_cerevisiae/sacCer3/all_chr.fa"
SRC_DIR="/n/projects/cm2363/bpreveal/src"
SCRATCH_DIR="/dev/shm"
```

```

NUM_CORS=7
#I'm using a smaller population size and generation count
# than I would use in real work. This is just a demo, after all!
POP_SIZE=1000
NUM_GENERATIONS=1000

with pysam.FastaFile(GENOME_FNAME) as genome:
    ORIG_SEQUENCE=genome.fetch("chrII", INPUT_START, INPUT_END + NUM_CORS)

    # Used for the PISA part.
    LONG_SEQUENCE= genome.fetch("chrII", INPUT_START,
                                INPUT_END+OUTPUT_LENGTH+NUM_CORS)

# There are a few motifs I want to annotate, as well as the Pho5 gene itself.
# I'm also putting up a box that reflects the area that my fitness
# function was trying to tamp down.
annotations = [
    ((OUTPUT_START+450, OUTPUT_START+550), "Targ", "violet"),
    ((431075, 431089), "FKH2", "goldenrod"),
    ((431200, 431206), "PH04", "teal"),
    ((431310, 431316), "PH04", "teal"),
    ((430700, 430951), "Pho5", "slateblue")]

```

```
[3]: predictor = utils.BatchPredictor(MODEL_FNAME, 32)
```

```
[4]: predictor.submitString(ORIG_SEQUENCE[:INPUT_LENGTH], 1)
      (origLogits, origLogcounts), label = predictor.getOutput()
```

```
[5]: #I want to get a list of corruptors that don't intersect any of my annotations
      ↪above.
      # (other than the target nucleosome, obviously.)
      # I'll also add an extra 3 bp of padding around the annotated motifs.
      annotPoses = [(x[0][0]-INPUT_START-3, x[0][1]-INPUT_START+3) for x in
      ↪annotations[1:]]

      initialCorList = gaOptimize.getCandidateCorruptorList(
          ORIG_SEQUENCE, regions=[[BUFFER, len(ORIG_SEQUENCE)-BUFFER]],
          allowDeletion=True, allowInsertion=True)
      print(initialCorList[:10])
      corList = []
      # I could figure out the regions that I want to allow mutations in, and then
      # use the regions parameter to getCandidateCorruptorList, but for removing
      # corruptors, it's honestly easier to just use a quick loop:
      for c in initialCorList:
          add = True
          for ap in annotPoses:
              if c[0] >= ap[0] and c[0] <= ap[1]:

```

```

        add = False
    if add:
        corList.append(c)
print(corList[:10])

```

```

[(1046, 'CGTdÄČĜĚ'), (1047, 'ACTdÄČĜĚ'), (1048, 'CGTdÄČĜĚ'), (1049, 'AGTdÄČĜĚ'),
(1050, 'ACGdÄČĜĚ'), (1051, 'ACTdÄČĜĚ'), (1052, 'CGTdÄČĜĚ'), (1053, 'AGTdÄČĜĚ'),
(1054, 'CGTdÄČĜĚ'), (1055, 'ACTdÄČĜĚ')]
[(1301, 'CGTdÄČĜĚ'), (1302, 'CGTdÄČĜĚ'), (1303, 'ACGdÄČĜĚ'), (1304, 'AGTdÄČĜĚ'),
(1305, 'ACGdÄČĜĚ'), (1306, 'AGTdÄČĜĚ'), (1307, 'ACTdÄČĜĚ'), (1308, 'CGTdÄČĜĚ'),
(1309, 'CGTdÄČĜĚ'), (1310, 'ACGdÄČĜĚ')]

```

```

[6]: # This is a function that runs PISA on a given Organism.
# It just writes the appropriate files out to disk, and then calls
# the BPREveal PISA script from the shell.
def runPisa(seqSource, name):
    # This calls the pisa script from BPREveal. It generates a fasta file
    # of the sequence with the listed mutations applied. It saves it to a
    # file with a given name so that you can reference it later. Of course,
    # this file is in temporary storage, so you'll have to regenerate it each
    # time you run the script. seqSource may be either a string (in which
    # case bases [BUFFER..BUFFER+OUTPUT_LENGTH]) will be analyzed, or an
    # Organism, in which case this function will call getSequence with
    # LONG_SEQUENCE, and analyze the bases in the output window.

    if(type(seqSource) == str):
        fullSeq = seqSource[:INPUT_LENGTH+OUTPUT_LENGTH]
    else:
        fullSeq = seqSource.getSequence(LONG_SEQUENCE,
        ↪INPUT_LENGTH+OUTPUT_LENGTH)
    with open(SCRATCH_DIR + "/pisa.fa", "w") as fastaFp:
        for pos in range(OUTPUT_LENGTH):
            seq = fullSeq[pos:pos+INPUT_LENGTH]
            fastaFp.write(">{0:d}\n".format(pos))
            fastaFp.write("{0:s}\n".format(seq))
    for i in range(2):
        jsonName = SCRATCH_DIR + "/pisa{0:d}.json".format([3,5][i])
        with open(jsonName, "w") as outJson:
            config = {"model-file" : MODEL_FNAME,
                    "sequence-fasta" : SCRATCH_DIR + "/pisa.fa",
                    "num-shuffles" : 20,
                    "head-id" : 0,
                    "task-id" : i,
                    "output-h5" : SCRATCH_DIR + "/" + name + "_pisa{0:d}.h5".
            ↪format([3,5][i]),
                    "output-length" : 1000,
                    "input-length" : INPUT_LENGTH,

```

```

        "verbosity" : "WARNING"}
    json.dump(config, outJson)
    !{SRC_DIR}/interpretPisaFasta.py {jsonName}

```

```

[7]: def loadPisa(fname, offset):
    with h5py.File(fname, "r") as fp:
        dats = np.sum(fp["shap"], axis=2)

    skewMat = np.zeros((dats.shape[0], dats.shape[1]+dats.shape[0]))
    for i in range(0, dats.shape[0]):
        if(i+offset < 0 or i+offset >=1000 ):
            continue
        skewMat[i+offset,i:i+dats.shape[1]] = dats[i]
    skewMat = skewMat[:,BUFFER:BUFFER+1000]
    return skewMat
def plotPisa(name, mutations, both=True, limit=None,
            profile=None, annotations=None):
    dat3 = loadPisa(SCRATCH_DIR + "/" + name + "_pisa3.h5", -OFFSET)
    dat5 = loadPisa(SCRATCH_DIR + "/" + name + "_pisa5.h5", OFFSET)
    if(both):
        skewMat = dat3 + dat5
    else:
        skewMat = dat5

    if(limit is None):
        limit = np.max(np.abs(skewMat))/3
    if annotations is not None:
        for annot in annotations:
            skewMat[:70,
                    annot[0][0]-OUTPUT_START:annot[0][1]-OUTPUT_START] = limit/2
    for mut in mutations:
        for row in range(1,OFFSET):
            startPos = mut[0]-row//10 - BUFFER
            stopPos = mut[0] + row//10+1 - BUFFER
            skewMat[-(OFFSET-row), startPos:stopPos] = -limit/2
    fig = plt.figure()

    axImg = fig.add_axes([0.1, 0.1, 0.7, 0.7])
    axImg.imshow(skewMat, vmin=-limit, vmax=limit,
                 cmap='RdBu_r', aspect='auto')
    axImg.set_xticks(range(0,1000,200), range(OUTPUT_START, OUTPUT_END, 200))
    axImg.set_yticks(range(0,1000,200), range(OUTPUT_START, OUTPUT_END, 200))

    axSum = fig.add_axes([0.8, 0.1, 0.1, 0.7])
    axSum.tick_params(bottom=False, labelbottom=False,
                     left=False, labelleft=False)
    if(profile is None):

```

```

sumDats = scipy.special.softmax(
    np.sum(skewMat, axis=1)[OFFSET:-OFFSET])
#If no profile values are given, compute one by summing the pisa plot.
axSum.plot(sumDats, range(OFFSET,1000-OFFSET), color='tab:gray')
axSum.set_xlim((-max(sumDats)/20, max(sumDats)))
else:
    #Plot both the 3' and 5' data.
    axSum.plot(profile[:,0], range(0,1000)[::-1], color='tab:orange')
    axSum.plot(profile[:,1], range(0,1000)[::-1], color='tab:blue')
    axSum.set_xlim((-np.max(profile)/20, np.max(profile)))
axSum.set_ylim((0,1000))

axImp = fig.add_axes([0.1, 0.8, 0.7, 0.1])
axImp.tick_params(bottom=False, labelbottom=False,
                  left=False, labelleft=False)
sumImp = np.sum(np.abs(skewMat[OFFSET:-OFFSET,:]), axis=0)
axImp.plot(range(0,1000), sumImp, color='tab:gray')
axImp.set_xlim((0, 1000))

return limit

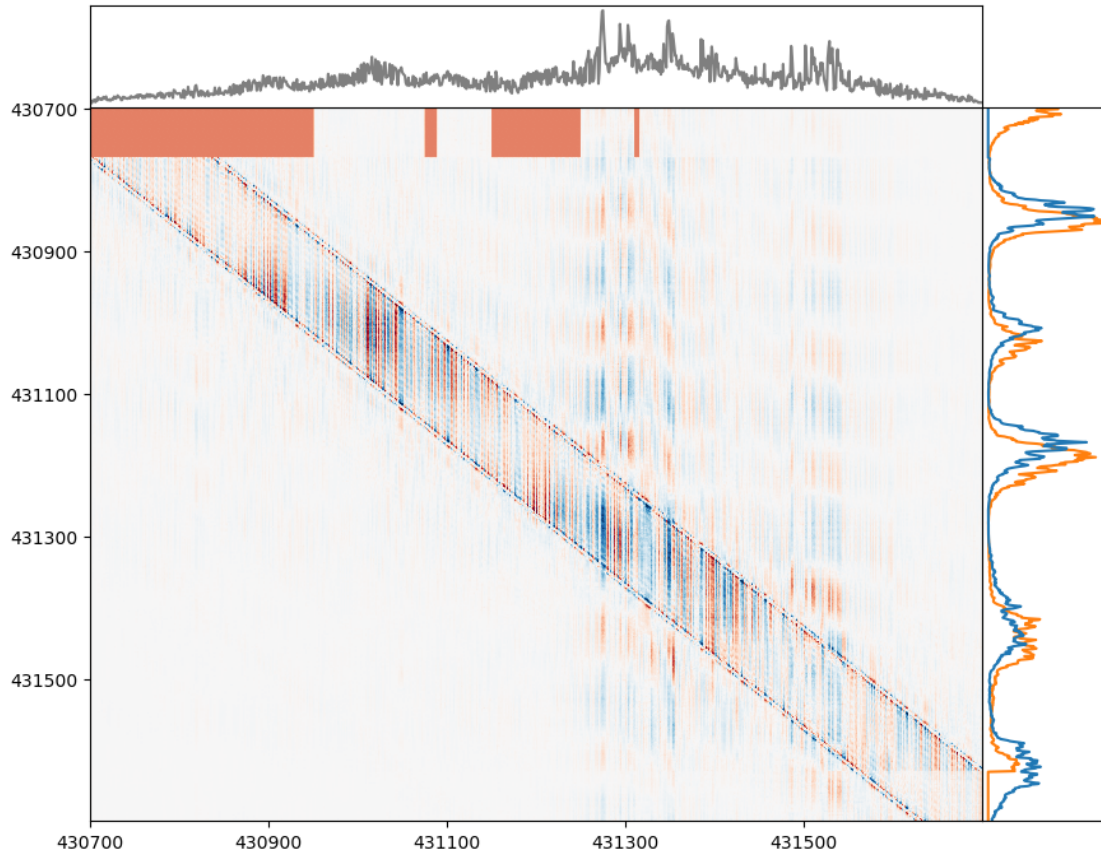
def skewProfile(origProfile):
    # This is very specific to the way I'm processing mnase data.
    # Since I train on 3' and 5' cut sites, you always have to
    # mentally shift the profiles over to where the dyad should be.
    # This function offsets the profiles by half a nucleosome
    # so that they line up with where the dyads should be.
    newProfile = np.zeros(origProfile.shape)
    newProfile[OFFSET:,1] = origProfile[:-OFFSET,1]
    newProfile[:-OFFSET,0] = origProfile[OFFSET:,0]
    return newProfile

```

```
[7]: #runPisa(LONG_SEQUENCE, "wt")
```

```
[8]: origProfile = utils.logitsToProfile(origLogits, origLogcounts)
origSkewProfile = skewProfile(origProfile)
plotPisa("wt", [], profile=origSkewProfile, annotations=annotations)
```

```
[8]: 0.23331515863537788
```



This PISA plot shows us how the nucleosomes are positioned in the wild type. The long-range bands around 431300 are in the NDR, which is interesting.

2 Studying the GA's outputs

Here, I'll run a quick optimization, and we'll look at what the GA invented in PISA space.

```
[9]: def fitnessSuppressSecond(preds, cors):
    logits, logcounts = preds
    profile = utils.logitsToProfile(logits, 0)
    profile = skewProfile(profile)
    targetArea = np.sum(profile[450:550,:])
    # Note that fitness should be large for good organisms, and low for
    # bad ones. This is the opposite of how loss is defined in neural networks.
    return 1 - targetArea
pop = gaOptimize.Population(ORIG_SEQUENCE, INPUT_LENGTH, POP_SIZE,
                           NUM_CORS, corList, lambda x:True,
                           fitnessSuppressSecond, 10, predictor)
```

```
[10]: pop.runCalculation()
recordScores = []
recordGenerations = []
recordProfiles = []
recordCorruptors = []
fitnessArray = []

# Now it's time to actually run the GA.
bestScore = -10000000
pbar = tqdm.tqdm(range(NUM_GENERATIONS))
for i in pbar:
    pop.nextGeneration()
    pop.runCalculation()
    allFitnesses = [x.score for x in pop.organisms]
    fitnessArray.append(allFitnesses)
    if(pop.organisms[-1].score > bestScore):
        recordProfiles.append(pop.organisms[-1].profile)
        bestScore = pop.organisms[-1].score
        recordScores.append(bestScore)
        recordGenerations.append(i)
        pbar.set_description("Gen {0:d} fitness {1:f}".format(i, bestScore))
```

Gen 965 fitness 0.979513: 100%|

| 1000/1000 [05:10<00:00, 3.22it/s]

```
[11]: print(pop.organisms[-1].corruptors)
```

```
[(1503, 'T'), (1510, 'T'), (1532, 'A'), (1564, 'A'), (1644, 'Č'), (1699, 'd'),
(1832, 'G')]
```

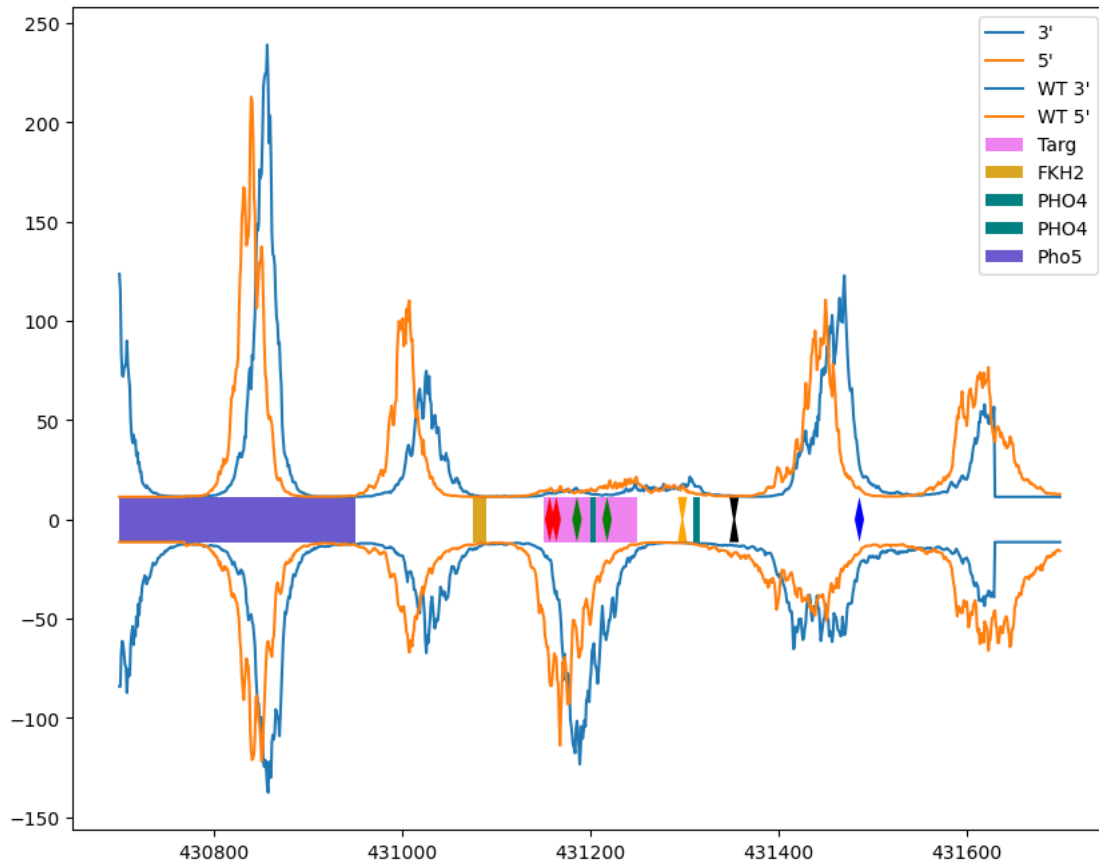
```
[12]: fig = plt.figure()
ax=fig.add_subplot()

# Get the actual profiles for the best organism and the original sequence.
logits, logcounts = pop.organisms[-1].profile
profile = skewProfile(utils.logitsToProfile(logits, logcounts))
origProfile = skewProfile(utils.logitsToProfile(origLogits, origLogcounts))
# In order to add pips for the corruptors, I need to have them at the
# correct X coordinates. The numbers in the corruptors are relative to the
# start of the input, and the plot will be relative to the genome itself.
cors = pop.organisms[-1].corruptors
corsRepositioned = [(x[0] + INPUT_START, x[1]) for x in cors]
# I'm shifting the profiles again so that we have a sense of where
```

```

# the midpoints would be.
offset = OFFSET*2
gaOptimize.plotTraces(
    [ (profile[:,0], "3'", "tab:blue"),
      (profile[:,1], "5'", "tab:orange")],
    [ (origProfile[:,0], "WT 3'", "tab:blue"),
      (origProfile[:,1], "WT 5'", "tab:orange")],
    range(OUTPUT_START, OUTPUT_END),
    annotations, corsRepositioned, ax)

```



```

[13]: s = pop.organisms[-1].getSequence(LONG_SEQUENCE, INPUT_LENGTH + OUTPUT_LENGTH)
      runPisa(s, "mut1")

```

```

0%|                               | 0/1000 [00:00<?,
?it/s]WARNING:tensorflow:From /scratch/bpreveal-teak/lib/python3.10/site-
packages/tensorflow/python/util/deprecation.py:561: calling function (from
tensorflow.python.eager.polymorphic_function.polymorphic_function) with
experimental_relax_shapes is deprecated and will be removed in a future version.
Instructions for updating:
experimental_relax_shapes is deprecated, use reduce_retracing instead

```



```
2023-06-21 10:13:24.190583: W tensorflow/c/c_api.cc:291] Operation
'{name:'AssignVariableOp_29' id:333 op device:{requested: '/device:CPU:0',
assigned: ''} def:{{{node AssignVariableOp_29}}} =
AssignVariableOp[_has_manual_control_dependencies=true, dtype=DT_FLOAT,
validate_shape=false, _device="/device:CPU:0"](kernel_9, Identity_29)}}' was
changed by setting attribute after it was run by a session. This mutation will
have no effect, and will trigger an error in the future. Either don't modify
nodes after running them or create a new session.
WARNING:tensorflow:From /n/projects/cm2363/bpreveal/src/shap.py:147: The name
tf.keras.backend.get_session is deprecated. Please use
tf.compat.v1.keras.backend.get_session instead.
```

```
/scratch/bpreveal-teak/lib/python3.10/site-
packages/keras/engine/training_v1.py:2357: UserWarning: `Model.state_updates`
will be removed in a future version. This property should not be used in
TensorFlow 2.0, as `updates` are applied automatically.
```

```
updates=self.state_updates,
2023-06-21 10:13:24.340778: W tensorflow/c/c_api.cc:291] Operation
'{name:'solo_profile_mnase/BiasAdd' id:592 op device:{requested: '', assigned:
''} def:{{{node solo_profile_mnase/BiasAdd}}} = BiasAdd[T=DT_FLOAT,
_has_manual_control_dependencies=true,
data_format="NHWC"](solo_profile_mnase/Conv1D/Squeeze,
solo_profile_mnase/BiasAdd/ReadVariableOp)}}' was changed by setting attribute
after it was run by a session. This mutation will have no effect, and will
trigger an error in the future. Either don't modify nodes after running them or
create a new session.
```

```
100%|          | 1000/1000 [00:34<00:00, 28.87it/s]
0%|          | 0/1000 [00:00<?,
?it/s]WARNING:tensorflow:From /scratch/bpreveal-teak/lib/python3.10/site-
packages/tensorflow/python/util/deprecation.py:561: calling function (from
tensorflow.python.eager.polymorphic_function.polymorphic_function) with
experimental_relax_shapes is deprecated and will be removed in a future version.
Instructions for updating:
```

```
experimental_relax_shapes is deprecated, use reduce_retracing instead
2023-06-21 10:13:59.119250: W tensorflow/c/c_api.cc:291] Operation
'{name:'AssignVariableOp_17' id:309 op device:{requested: '/device:CPU:0',
assigned: ''} def:{{{node AssignVariableOp_17}}} =
AssignVariableOp[_has_manual_control_dependencies=true, dtype=DT_FLOAT,
validate_shape=false, _device="/device:CPU:0"](kernel_3, Identity_17)}}' was
changed by setting attribute after it was run by a session. This mutation will
have no effect, and will trigger an error in the future. Either don't modify
nodes after running them or create a new session.
WARNING:tensorflow:From /n/projects/cm2363/bpreveal/src/shap.py:147: The name
tf.keras.backend.get_session is deprecated. Please use
tf.compat.v1.keras.backend.get_session instead.
```

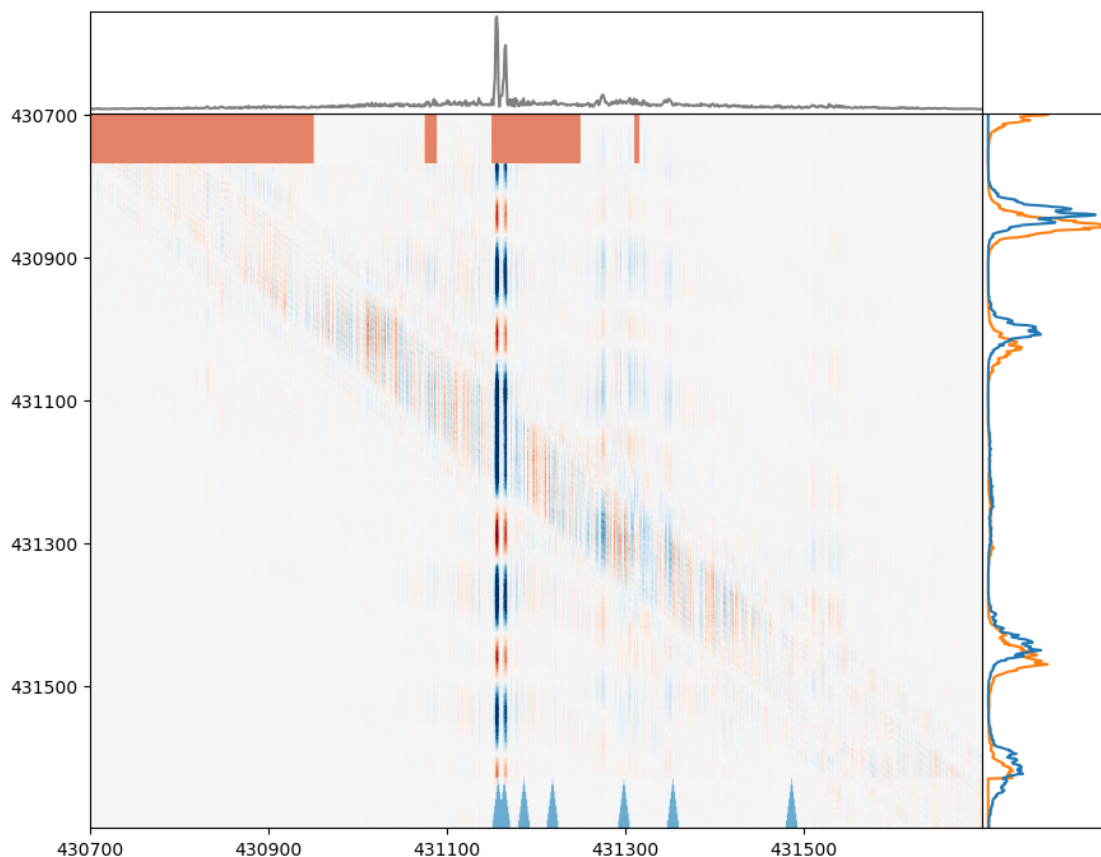
```
/scratch/bpreveal-teak/lib/python3.10/site-
packages/keras/engine/training_v1.py:2357: UserWarning: `Model.state_updates`
```

will be removed in a future version. This property should not be used in TensorFlow 2.0, as `updates` are applied automatically.

```
updates=self.state_updates,  
2023-06-21 10:13:59.256750: W tensorflow/c/c_api.cc:291] Operation  
'{name:'solo_profile_mnase/BiasAdd' id:592 op device:{requested: '', assigned:  
''} def:{{{node solo_profile_mnase/BiasAdd}}} = BiasAdd[T=DT_FLOAT,  
_has_manual_control_dependencies=true,  
data_format="NHWC"](solo_profile_mnase/Conv1D/Squeeze,  
solo_profile_mnase/BiasAdd/ReadVariableOp)}}' was changed by setting attribute  
after it was run by a session. This mutation will have no effect, and will  
trigger an error in the future. Either don't modify nodes after running them or  
create a new session.  
100%|                                     | 1000/1000 [00:34<00:00, 28.86it/s]
```

```
[14]: plotPisa("mut1", pop.organisms[-1].corruptors, profile=profile,  
          annotations=annotations)
```

```
[14]: 0.41989924510320026
```



3 The problem with the solution

That looks good, and it's cool how the algorithm discovered new motifs to add to get the desired effect, but it's got a big problem. The issue is that the other nucleosomes moved, and I don't want to mess with them. Instead, I want those nucleosomes to stay where they were. I'm going to modify my fitness function, so that in addition to rewarding the GA for decreasing nucleosome density in the middle, I also reward it for being correlated with the original profile outside of the N-2 zone.

```
[15]: import scipy
remOld = np.ravel(np.concatenate([origSkewProfile[:450,:], origSkewProfile[550:
↪, :]]))
def fitnessSuppressSecondCorrelate(preds, cors):
    logits, logcounts = preds
    profile = utils.logitsToProfile(logits, 0)
    profile = skewProfile(profile)
    totalArea = np.sum(profile[450:550,:])
    remNew = np.ravel(np.concatenate([profile[:450,:], profile[550:,:]]))
    corr = scipy.stats.pearsonr(remNew, remOld).statistic
    # Note that fitness should be large for good organisms, and low for
    # bad ones. This is the opposite of how loss is defined in neural networks.
    return 1 - totalArea + corr

popCor = gaOptimize.Population(ORIG_SEQUENCE, INPUT_LENGTH, POP_SIZE,
                                NUM_CORS, corList, lambda x: True,
                                fitnessSuppressSecondCorrelate, 10, predictor)
```

```
[16]: popCor.runCalculation()
recordScoresCor = []
recordGenerationsCor = []
recordProfilesCor = []
recordCorruptorsCor = []
fitnessArrayCor = []

# Now it's time to actually run the GA.
bestScore = -10000000
pbar = tqdm.tqdm(range(NUM_GENERATIONS))
for i in pbar:
    popCor.nextGeneration()
    popCor.runCalculation()
    allFitnessesCor = [x.score for x in popCor.organisms]
    fitnessArrayCor.append(allFitnessesCor)
    if(popCor.organisms[-1].score > bestScore):
        recordProfilesCor.append(popCor.organisms[-1].profile)
        bestScore = popCor.organisms[-1].score
        recordScoresCor.append(bestScore)
        recordGenerationsCor.append(i)
```

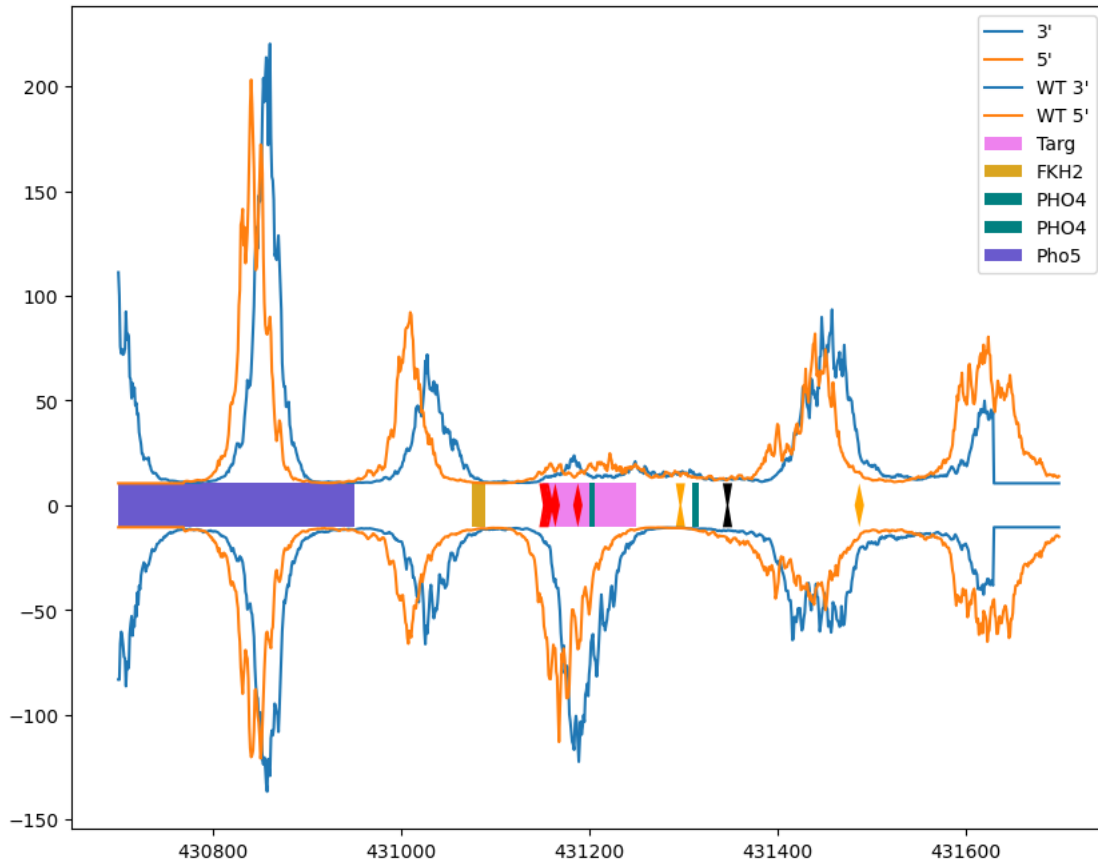
```
pbar.set_description("Gen {0:d} fitness {1:f}".format(i, bestScore))
```

```
Gen 926 fitness 1.925754: 100%|
```

```
| 1000/1000 [11:06<00:00, 1.50it/s]
```

```
[17]: fig = plt.figure()
      ax=fig.add_subplot()

      # Get the actual profiles for the best organism and the original sequence.
      logits, logcounts = popCor.organisms[-1].profile
      profile = skewProfile(utils.logitsToProfile(logits, logcounts))
      origProfile = skewProfile(utils.logitsToProfile(origLogits, origLogcounts))
      # In order to add pips for the corruptors, I need to have them at the
      # correct X coordinates. The numbers in the corruptors are relative to the
      # start of the input, and the plot will be relative to the genome itself.
      cors = popCor.organisms[-1].corruptors
      corsRepositioned = [(x[0] + INPUT_START, x[1]) for x in cors]
      # I'm shifting the profiles again so that we have a sense of where
      # the midpoints would be.
      offset = OFFSET*2
      gaOptimize.plotTraces(
          [ (profile[:,0], "3'", "tab:blue"),
            (profile[:,1], "5'", "tab:orange")],
          [ (origProfile[:,0], "WT 3'", "tab:blue"),
            (origProfile[:,1], "WT 5'", "tab:orange")],
          range(OUTPUT_START, OUTPUT_END),
          annotations, corsRepositioned, ax)
```



```
[18]: s = popCor.organisms[-1].getSequence(LONG_SEQUENCE, INPUT_LENGTH +
      ↪OUTPUT_LENGTH)
      runPisa(s, "mut1Cor")
```

```
0%|                               | 0/1000 [00:00<?,
?it/s]WARNING:tensorflow:From /scratch/bpreveal-teak/lib/python3.10/site-
packages/tensorflow/python/util/deprecation.py:561: calling function (from
tensorflow.python.eager.polymorphic_function.polymorphic_function) with
experimental_relax_shapes is deprecated and will be removed in a future version.
Instructions for updating:
experimental_relax_shapes is deprecated, use reduce_retracing instead
2023-06-21 10:25:42.533652: W tensorflow/c/c_api.cc:291] Operation
'{name:'AssignVariableOp_3' id:281 op device:{requested: '/device:CPU:0',
assigned: ''} def:{{{node AssignVariableOp_3}} =
AssignVariableOp[_has_manual_control_dependencies=true, dtype=DT_FLOAT,
validate_shape=false, _device="/device:CPU:0"](total_1, Identity_3)}}' was
changed by setting attribute after it was run by a session. This mutation will
have no effect, and will trigger an error in the future. Either don't modify
nodes after running them or create a new session.
WARNING:tensorflow:From /n/projects/cm2363/bpreveal/src/shap.py:147: The name
```

tf.keras.backend.get_session is deprecated. Please use
tf.compat.v1.keras.backend.get_session instead.

```
/scratch/bpreveal-teak/lib/python3.10/site-  
packages/keras/engine/training_v1.py:2357: UserWarning: `Model.state_updates`  
will be removed in a future version. This property should not be used in  
TensorFlow 2.0, as `updates` are applied automatically.  
    updates=self.state_updates,  
2023-06-21 10:25:42.671760: W tensorflow/c/c_api.cc:291] Operation  
'{name:'solo_profile_mnase/BiasAdd' id:592 op device:{requested: '', assigned:  
''} def:{{{node solo_profile_mnase/BiasAdd}}} = BiasAdd[T=DT_FLOAT,  
_has_manual_control_dependencies=true,  
data_format="NHWC"](solo_profile_mnase/Conv1D/Squeeze,  
solo_profile_mnase/BiasAdd/ReadVariableOp)}}' was changed by setting attribute  
after it was run by a session. This mutation will have no effect, and will  
trigger an error in the future. Either don't modify nodes after running them or  
create a new session.  
100%|          | 1000/1000 [00:34<00:00, 29.07it/s]  
0%|          | 0/1000 [00:00<?,  
?it/s]WARNING:tensorflow:From /scratch/bpreveal-teak/lib/python3.10/site-  
packages/tensorflow/python/util/deprecation.py:561: calling function (from  
tensorflow.python.eager.polymorphic_function.polymorphic_function) with  
experimental_relax_shapes is deprecated and will be removed in a future version.  
Instructions for updating:  
experimental_relax_shapes is deprecated, use reduce_retracing instead  
2023-06-21 10:26:17.240305: W tensorflow/c/c_api.cc:291] Operation  
'{name:'AssignVariableOp_28' id:331 op device:{requested: '/device:CPU:0',  
assigned: ''} def:{{{node AssignVariableOp_28}}} =  
AssignVariableOp[_has_manual_control_dependencies=true, dtype=DT_FLOAT,  
validate_shape=false, _device="/device:CPU:0"](bias_9, Identity_28)}}' was  
changed by setting attribute after it was run by a session. This mutation will  
have no effect, and will trigger an error in the future. Either don't modify  
nodes after running them or create a new session.  
WARNING:tensorflow:From /n/projects/cm2363/bpreveal/src/shap.py:147: The name  
tf.keras.backend.get_session is deprecated. Please use  
tf.compat.v1.keras.backend.get_session instead.
```

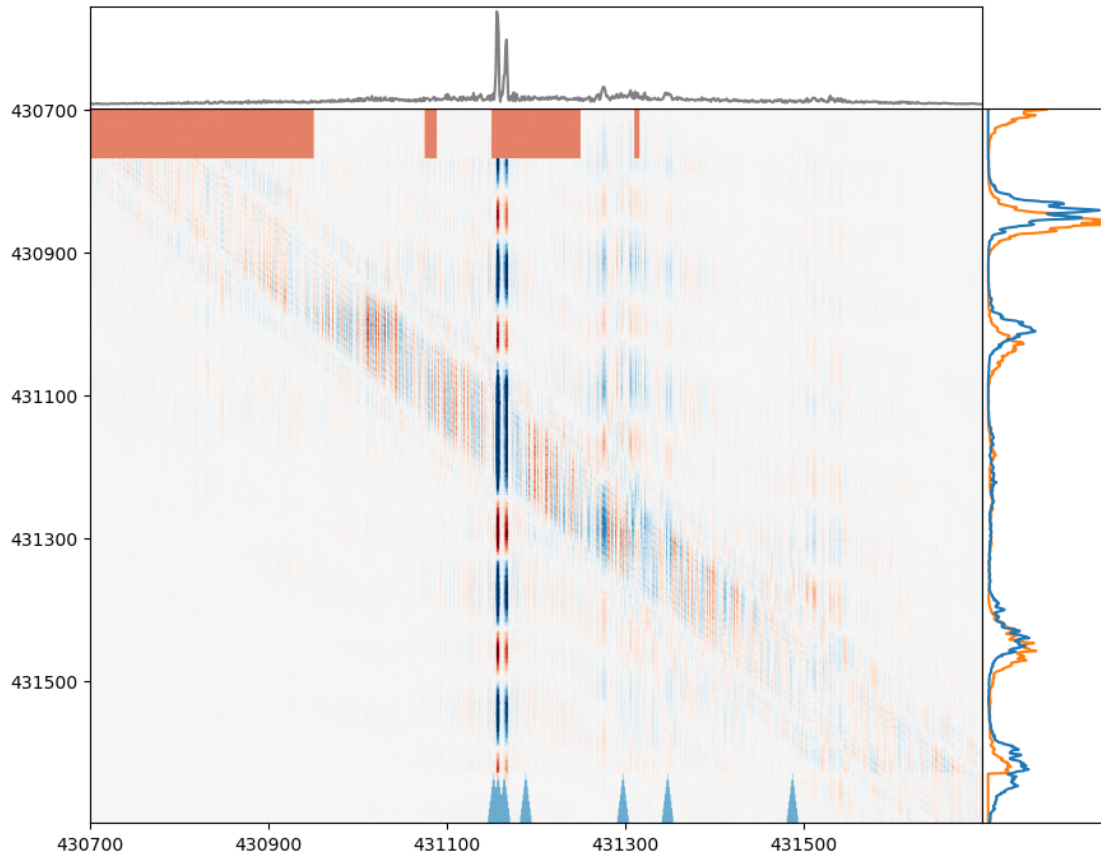
```
/scratch/bpreveal-teak/lib/python3.10/site-  
packages/keras/engine/training_v1.py:2357: UserWarning: `Model.state_updates`  
will be removed in a future version. This property should not be used in  
TensorFlow 2.0, as `updates` are applied automatically.  
    updates=self.state_updates,  
2023-06-21 10:26:17.375078: W tensorflow/c/c_api.cc:291] Operation  
'{name:'solo_profile_mnase/BiasAdd' id:592 op device:{requested: '', assigned:  
''} def:{{{node solo_profile_mnase/BiasAdd}}} = BiasAdd[T=DT_FLOAT,  
_has_manual_control_dependencies=true,  
data_format="NHWC"](solo_profile_mnase/Conv1D/Squeeze,  
solo_profile_mnase/BiasAdd/ReadVariableOp)}}' was changed by setting attribute
```

after it was run by a session. This mutation will have no effect, and will trigger an error in the future. Either don't modify nodes after running them or create a new session.

100%| | 1000/1000 [00:34<00:00, 29.14it/s]

```
[19]: plotPisa("mut1Cor", popCor.organisms[-1].corruptors, profile=profile,
          annotations=annotations)
```

[19]: 0.314358115196228



```
[20]: plt.subplot(121)
plt.imshow(fitnessArray, aspect='auto')
plt.colorbar()
# I'll also zoom in on the 20 best organisms, since it's hard to see
# the very rightmost pixels.
plt.subplot(122)
plt.imshow(np.array(fitnessArray)[:,-20:], aspect='auto', cmap='prism',
           ↪ interpolation='nearest')
plt.colorbar()
```

[20]: <matplotlib.colorbar.Colorbar at 0x7fc4008c1d80>

