Problem2

April 14, 2021

1 Problem 2

Predict the CO2 concentration of the Mount Lua Observatory using tensorflow probability time series modelling.

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import tensorflow_probability as tfp

from tensorflow_probability import distributions as tfd
from tensorflow_probability import sts

import collections
import matplotlib.dates as mdates
```

1.1 The Data

From the Mauna Loa Observatory

[2]:
```
# CO2 readings from Mauna Loa observatory, monthly beginning January 1966
# Original source: http://scrippsco2.ucsd.edu/data/atmospheric_co2/
↪primary_mlo_co2_record
```

```
co2_by_month = np.array('320.62,321.60,322.39,323.70,324.08,323.75,322.38,320.
↪36,318.64,318.10,319.78,321.03,322.33,322.50,323.04,324.42,325.00,324.09,322.
↪54,320.92,319.25,319.39,320.73,321.96,322.57,323.15,323.89,325.02,325.57,325.
↪36,324.14,322.11,320.33,320.25,321.32,322.89,324.00,324.42,325.63,326.66,327.
↪38,326.71,325.88,323.66,322.38,321.78,322.85,324.12,325.06,325.98,326.93,328.
↪14,328.08,327.67,326.34,324.69,323.10,323.06,324.01,325.13,326.17,326.68,327.
↪17,327.79,328.92,328.57,327.36,325.43,323.36,323.56,324.80,326.01,326.77,327.
↪63,327.75,329.73,330.07,329.09,328.04,326.32,324.84,325.20,326.50,327.55,328.
↪55,329.56,330.30,331.50,332.48,332.07,330.87,329.31,327.51,327.18,328.16,328.
↪64,329.35,330.71,331.48,332.65,333.09,332.25,331.18,329.39,327.43,327.37,328.
↪46,329.57,330.40,331.40,332.04,333.31,333.97,333.60,331.90,330.06,328.56,328.
↪34,329.49,330.76,331.75,332.56,333.50,334.58,334.88,334.33,333.05,330.94,329.
↪30,328.94,330.31,331.68,332.93,333.42,334.70,336.07,336.75,336.27,334.92,332.
↪75,331.59,331.16,332.40,333.85,334.97,335.38,336.64,337.76,338.01,337.89,336.
↪54,334.68,332.76,332.55,333.92,334.95,336.23,336.76,337.96,338.88,339.47,339.
↪29,337.73,336.09,333.92,333.86,335.29,336.73,338.01,338.36,340.07,340.77,341.
↪47,341.17,339.56,337.60,335.88,336.02,337.10,338.21,339.24,340.48,341.38,342.
↪51,342.91,342.25,340.49,338.43,336.69,336.86,338.36,339.61,340.75,341.61,342.
↪70,343.57,344.14,343.35,342.06,339.81,337.98,337.86,339.26,340.49,341.38,342.
↪52,343.10,344.94,345.76,345.32,343.98,342.38,339.87,339.99,341.15,342.99,343.
↪70,344.50,345.28,347.06,347.43,346.80,345.39,343.28,341.07,341.35,342.98,344.
↪22,344.97,345.99,347.42,348.35,348.93,348.25,346.56,344.67,343.09,342.80,344.
↪24,345.56,346.30,346.95,347.85,349.55,350.21,349.55,347.94,345.90,344.85,344.
↪17,345.66,346.90,348.02,348.48,349.42,350.99,351.85,351.26,349.51,348.10,346.
↪45,346.36,347.81,348.96,350.43,351.73,352.22,353.59,354.22,353.79,352.38,350.
↪43,348.73,348.88,350.07,351.34,352.76,353.07,353.68,355.42,355.67,355.12,353.
↪90,351.67,349.80,349.99,351.30,352.52,353.66,354.70,355.38,356.20,357.16,356.
```

The below block of code formats the data properly for computation.

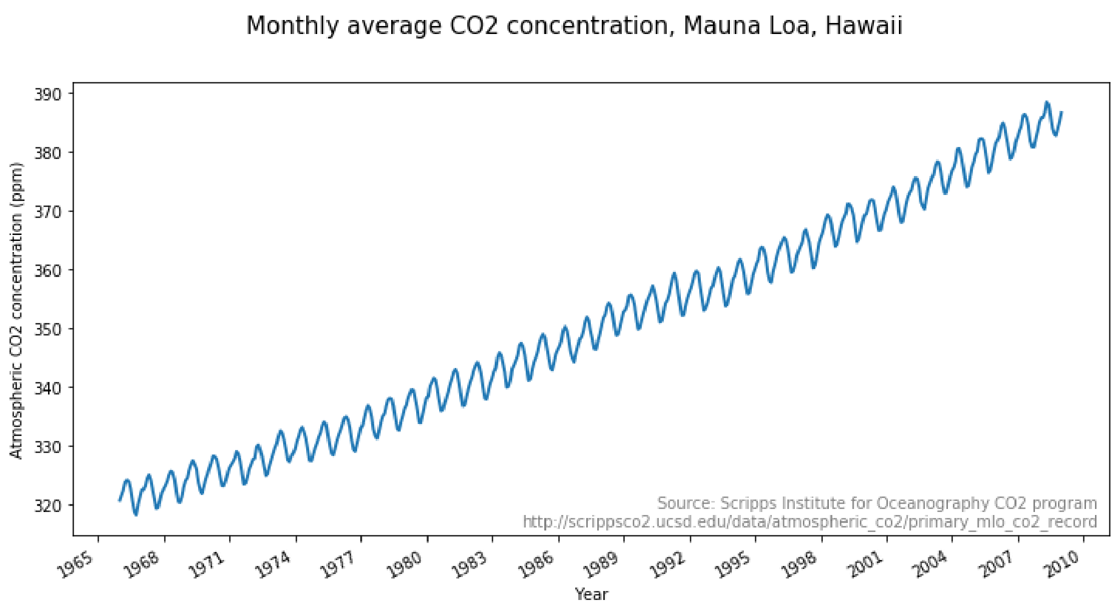
[3]:

```
co2_by_month = co2_by_month
num_forecast_steps = 12 * 10 # Forecast the final ten years, given previous data
co2_by_month_training_data = co2_by_month[:-num_forecast_steps]

co2_dates = np.arange("1966-01", "2019-02", dtype="datetime64[M]")
co2_loc = mdates.YearLocator(3)
co2_fmt = mdates.DateFormatter('%Y')
```

```
↪78,362.43,364.28,365.33,366.15,367.31,368.61,369.30,368.88,367.64,365.78,363.
↪90,364.23,365.46,366.97,368.15,368.87,369.59,371.14,371.00,370.35,369.27,366.
↪93,364.64,365.13,366.68,368.00,369.14,369.46,370.51,371.66,371.83,371.69,370.
↪12,368.12,366.62,366.73,368.29,369.53,370.28,371.50,372.12,372.86,374.02,373.
↪31,371.62,369.55,367.96,368.09,369.68,371.24,372.44,373.08,373.52,374.85,375.
↪55,375.40,374.02,371.48,370.70,370.25,372.08,373.78,374.68,375.62,376.11,377.
↪65,378.35,378.13,376.61,374.48,372.98,373.00,374.35,375.69,376.79,377.36,378.
```

Below, the monthly average CO2 concentration from the Mauna Loa Laboratory data is plotted.

[4]:
```
fig = plt.figure(figsize=(12, 6))
ax = fig.add_subplot(1, 1, 1)
ax.plot(co2_dates[:-num_forecast_steps], co2_by_month_training_data, lw=2,␣
↪label="training data")
ax.xaxis.set_major_locator(co2_loc)
ax.xaxis.set_major_formatter(co2_fmt)
ax.set_ylabel("Atmospheric CO2 concentration (ppm)")
ax.set_xlabel("Year")
fig.suptitle("Monthly average CO2 concentration, Mauna Loa, Hawaii",
             fontsize=15)
ax.text(0.99, .02,
        "Source: Scripps Institute for Oceanography CO2 program\nhttp://
↪scrippsco2.ucsd.edu/data/atmospheric_co2/primary_mlo_co2_record",
        transform=ax.transAxes,
        horizontalalignment="right",
        alpha=0.5)
fig.autofmt_xdate()
```



1.2 Model and Fitting

Below, the model is constructed.

[5]:
```
def build_model(observed_time_series):
  trend = sts.LocalLinearTrend(observed_time_series=observed_time_series)
  seasonal = tfp.sts.Seasonal(
      num_seasons=12, observed_time_series=observed_time_series)
  model = sts.Sum([trend, seasonal], observed_time_series=observed_time_series)
```

```
  return model
```

[6]:
```
co2_model = build_model(co2_by_month_training_data)

# Build the variational surrogate posteriors `qs`.
variational_posteriors = tfp.sts.build_factored_surrogate_posterior(
    model=co2_model)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-
packages/tensorflow/python/ops/linalg/linear_operator_composition.py:181:
LinearOperator.graph_parents (from tensorflow.python.ops.linalg.linear_operator)
is deprecated and will be removed in a future version.
Instructions for updating:
Do not call `graph_parents`.
WARNING:tensorflow:From /home/pi/.local/lib/python3.7/site-
packages/tensorflow_probability/python/distributions/distribution.py:298:
MultivariateNormalFullCovariance.__init__ (from
tensorflow_probability.python.distributions.mvn_full_covariance) is deprecated
and will be removed after 2019-12-01.
Instructions for updating:
`MultivariateNormalFullCovariance` is deprecated, use
`MultivariateNormalTriL(loc=loc,
scale_tril=tf.linalg.cholesky(covariance_matrix))` instead.
```

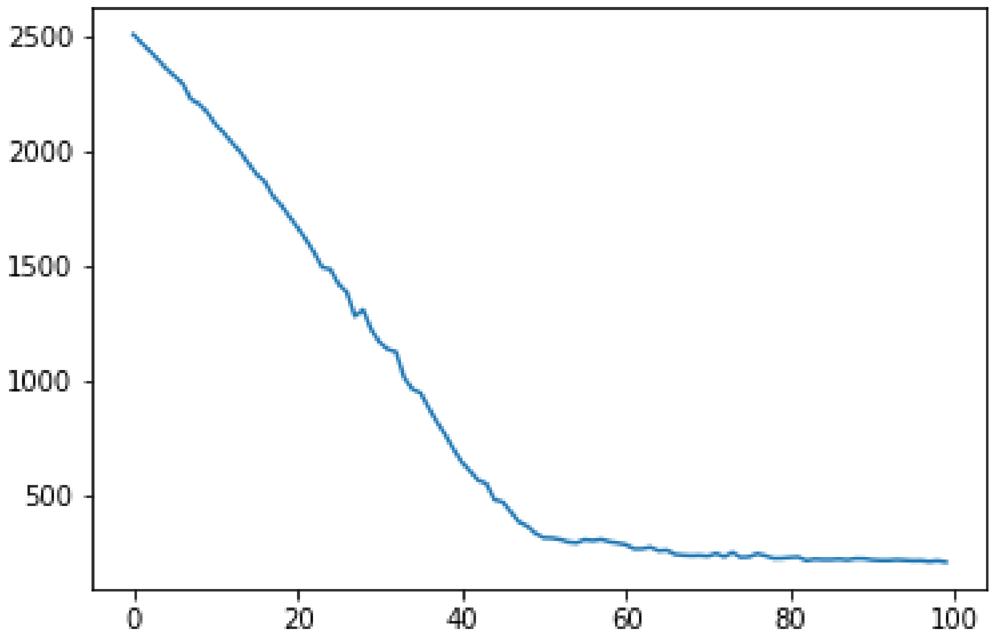
[7]:
```
# Allow external control of optimization to reduce test runtimes.
num_variational_steps = 100 # @param { isTemplate: true}
num_variational_steps = int(num_variational_steps)

optimizer = tf.optimizers.Adam(learning_rate=.1)
```

[8]:
```
def train():
    target_log_prob_fn=co2_model.joint_log_prob(
        observed_time_series=co2_by_month_training_data)
    elbo_loss_curve = tfp.vi.fit_surrogate_posterior(
        target_log_prob_fn=target_log_prob_fn,
        surrogate_posterior=variational_posteriors,
        optimizer=optimizer,
        num_steps=num_variational_steps)
    return elbo_loss_curve
```

[9]:
```
elbo_loss_curve = train()

plt.plot(elbo_loss_curve)
ax.set(title='ELBO Loss Curve', xlabel='Iteration', ylabel='Lose');
plt.show()

# Draw samples from the variational posterior.
```

```
q_samples_co2_ = variational_posteriors.sample(50)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-
packages/tensorflow/python/ops/linalg/linear_operator_diag.py:175: calling
LinearOperator.__init__ (from tensorflow.python.ops.linalg.linear_operator) with
graph_parents is deprecated and will be removed in a future version.
Instructions for updating:
Do not pass `graph_parents`.  They will  no longer be used.
```



[10]:
```
print("Inferred parameters:")
for param in co2_model.parameters:
    print("{}: {} +- {}".format(param.name,
                                np.mean(q_samples_co2_[param.name], axis=0),
                                np.std(q_samples_co2_[param.name], axis=0)))
```

```
Inferred parameters:
observation_noise_scale: 0.19187507033348083 +- 0.012764197774231434
LocalLinearTrend/_level_scale: 0.14980974793434143 +- 0.02310906909406185
LocalLinearTrend/_slope_scale: 0.030756875872612 +- 0.001747949281707406
Seasonal/_drift_scale: 0.03882041573524475 +- 0.007907980121672153
```

1.3 Predictions

[11]:
```
co2_forecast_dist = tfp.sts.forecast(
    co2_model,
    observed_time_series=co2_by_month_training_data,
    parameter_samples=q_samples_co2_,
    num_steps_forecast=num_forecast_steps)
```

[12]:
```
num_samples=10

co2_forecast_mean, co2_forecast_scale, co2_forecast_samples = (
    co2_forecast_dist.mean().numpy()[..., 0],
    co2_forecast_dist.stddev().numpy()[..., 0],
    co2_forecast_dist.sample(num_samples).numpy()[..., 0])
```

[13]:
```
def plot_forecast(x, y,
                  forecast_mean, forecast_scale, forecast_samples,
                     title, x_locator=None, x_formatter=None):
  """Plot a forecast distribution against the 'true' time series."""
  c1, c2 = 'g', 'tab:cyan'
  fig = plt.figure(figsize=(12, 6))
  ax = fig.add_subplot(1, 1, 1)

  num_steps = len(y)
  num_steps_forecast = forecast_mean.shape[-1]
  num_steps_train = num_steps - num_steps_forecast


  ax.plot(x, y, lw=2, color=c1, label='ground truth')

  forecast_steps = np.arange(
      x[num_steps_train],
      x[num_steps_train]+num_steps_forecast,
      dtype=x.dtype)

  ax.plot(forecast_steps, forecast_samples.T, lw=1, color=c2, alpha=0.1)

  ax.plot(forecast_steps, forecast_mean, lw=2, ls='--', color=c2,
           label='forecast')
  ax.fill_between(forecast_steps,
                   forecast_mean-2*forecast_scale,
                   forecast_mean+2*forecast_scale, color=c2, alpha=0.2)

  ymin, ymax = min(np.min(forecast_samples), np.min(y)), max(np.
↪max(forecast_samples), np.max(y))
  yrange = ymax-ymin
  ax.set_ylim([ymin - yrange*0.1, ymax + yrange*0.1])
```

```python
  ax.set_title("{}".format(title))
  ax.legend()

  if x_locator is not None:
    ax.xaxis.set_major_locator(x_locator)
    ax.xaxis.set_major_formatter(x_formatter)
    fig.autofmt_xdate()

  return fig, ax
```

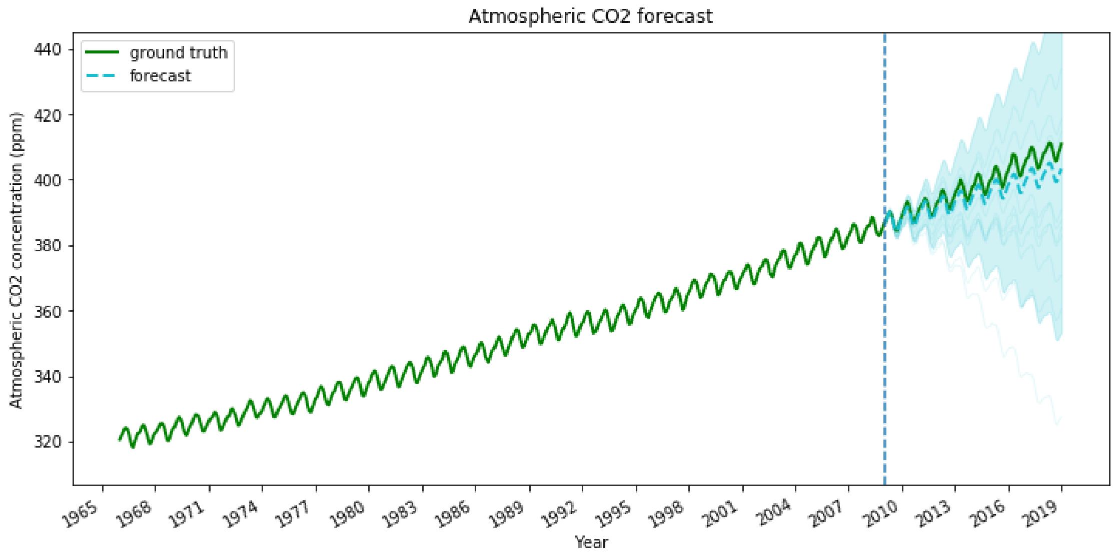
[14]:

```python
def plot_components(dates,
                    component_means_dict,
                    component_stddevs_dict,
                    x_locator=None,
                    x_formatter=None):
  """Plot the contributions of posterior components in a single figure."""
  c1, c2 = 'g', 'tab:cyan'

  axes_dict = collections.OrderedDict()
  num_components = len(component_means_dict)
  fig = plt.figure(figsize=(12, 2.5 * num_components))
  for i, component_name in enumerate(component_means_dict.keys()):
    component_mean = component_means_dict[component_name]
    component_stddev = component_stddevs_dict[component_name]

    ax = fig.add_subplot(num_components,1,1+i)
    ax.plot(dates, component_mean, lw=2)
    ax.fill_between(dates,
                     component_mean-2*component_stddev,
                     component_mean+2*component_stddev,
                     color=c2, alpha=0.5)
    ax.set_title(component_name)
    if x_locator is not None:
      ax.xaxis.set_major_locator(x_locator)
      ax.xaxis.set_major_formatter(x_formatter)
    axes_dict[component_name] = ax
  fig.autofmt_xdate()
  fig.tight_layout()
  return fig, axes_dict
```

[15]:

```python
fig, ax = plot_forecast(
    co2_dates, co2_by_month,
    co2_forecast_mean, co2_forecast_scale, co2_forecast_samples,
    x_locator=co2_loc,
    x_formatter=co2_fmt,
    title="Atmospheric CO2 forecast")
ax.axvline(co2_dates[-num_forecast_steps], linestyle="--")
```

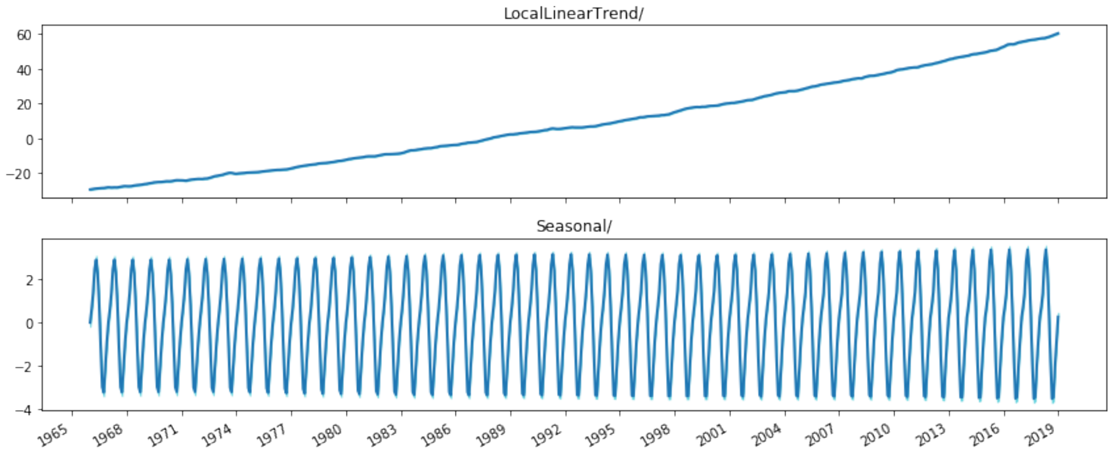
```python
ax.legend(loc="upper left")
ax.set_ylabel("Atmospheric CO2 concentration (ppm)")
ax.set_xlabel("Year")
fig.autofmt_xdate()
```



[16]:
```python
# Build a dict mapping components to distributions over
# their contribution to the observed signal.
component_dists = sts.decompose_by_component(
    co2_model,
    observed_time_series=co2_by_month,
    parameter_samples=q_samples_co2_)
```

[17]:
```python
co2_component_means_, co2_component_stddevs_ = (
    {k.name: c.mean() for k, c in component_dists.items()},
    {k.name: c.stddev() for k, c in component_dists.items()})
```

[18]:
```python
_ = plot_components(co2_dates, co2_component_means_, co2_component_stddevs_,
                    x_locator=co2_loc, x_formatter=co2_fmt)
```



1.4 References

Article this work is based off of: https://blog.tensorflow.org/2019/03/structural-time-series-modeling-in.html

Github code this is based off of: https://github.com/tensorflow/probability/blob/master/tensorflow_probability/e

```
[ ]:
```