Lab 2: Modeling Stellar Spectra

Astro 128 / 256 (UC Berkeley)

Feb 25, 2019

The main goal of this lab is to (a) build a generative model to predict what a stellar spectrum should look like for a given set of stellar properties (e.g., temperature, surface gravity, and elemental abundances; we'll refer to these properties as "labels"), and (b) use this model to infer the properties of stars by fitting their spectra.

We'll be using spectra of red giant stars from the APOGEE survey. Our spectral model will be *data-driven*. This means that we'll start with a *training set* of spectra whose labels are known a priori, and use these spectra to learn how the spectrum varies with each label. The paper Ness et al. 2015 describes the procedure we'll use in detail and should serve as a useful reference throughout. We recommend reading it. We also recommend reading Majewski et al. 2017, which describes the APOGEE survey, and Holtzman et al. 2015, which describes the pipeline APOGEE uses to fit labels from spectra.

a) Download a subset of the APOGEE spectra in the form of apStar files, following the instructions at this link. APOGEE spectra are sorted into different directories on the SDSS server based on their "Location ID", a 4 digit number. For this lab, we'll download all the spectra with the following 4 (randomly chosen) "location ids": 4230, 4262, 5162, 4241. This will give us about 3000 spectra. Figuring out how to efficiently download and access the spectra is part of the lab.

Each spectrum should be in its own apStar ".fits" file, with a name like "apStar-r8-2M19395890+2258341.fits". Each apStar file contains individual visit and coadded multi-visit spectra for one star. Explain briefly what this means, and what the point of multi-vist spectra is.

Helpfully, the coadded spectra have already been Doppler shifted to rest frame. Explain briefly what this means, and why the Doppler shift will be different for each visit. The apStar files also contain the associated error arrays and a quality flag bitmask, plus some other information. The data model is described in detail on the SDSS website. Figure out how to read in each spectrum and reconstruct the wavelength array. Plot an example spectrum or two. What are the units of spectra? Explain briefly what these units mean.

b) To build a training set, we also need to know the stellar properties ("labels") that have been derived for each spectrum by the ASPCAP pipeline (Garcia-Perez et al. 2015). These can be found in the "allStar" catalog, which you should download. For each spectrum you downloaded, find its labels in the allStar catalog.

Due to data quality issues, not all labels have been derived for all stars. Discard all spectra for which any of the following labels have not been derived: Teff, logg, [Fe/H], [Mg/Fe], [Si/Fe]. Also discard spectra with low signal-to-noise ratio (SNR < 50, as reported in the allStar catalog) or low metallicity ([Fe/H] < -1). This should leave you with roughly 1700 spectra.

Visualize the distribution of stars in label space using a corner plot.

c) Before we fit spectra, we need to "pseudo-continuum normalize" them, as described in Ness et al. 2015. Explain why this is necessary, and then normalize your spectra and error arrays. You can either write your own function to do this, or use a normalization function from the publicly available APOGEE package written by Jo Bovy.

Plot an example un-normalized spectrum, the derived pseudo-continuum, and the normalized version of the spectrum.

d) Use the apStar bitmasks to identify bad pixels in each spectrum (i.e. pixels where sky subtraction failed, there was a cosmic ray strike, or something else bad happened). Set the uncertainty in these pixels to a large value, so that they will not contribute significantly to the likelihood function in your fitting.

e) Now that you have cleaned spectra, divide them into two randomly selected groups of roughly equal size. Designate one group the training set and the other the cross-validation set.

f) Now for the meat of this problem. Use the training set to build a spectral model that predicts the spectrum *at each wavelength pixel* as a function of the following 5 labels: Teff, logg, [Fe/H], [Mg/Fe], [Si/Fe]. The details of the model (what functional form, how many free parameters, etc.) are up to you. Ness et al. 2015 find that a 2nd-order polynomial in labels work well, but any model is fine if it works. (I.e., if it can account for most of the variance within the training set). Your final spectral model will consist of thousands of individual models – one for each wavelength pixel – stitched together. You may find it useful to:

- rescale the labels such that they are all of order unity
- fit for a "scatter" term at each wavelength pixel to allow for some bad pixels or pixels with incorrect uncertainties.
- downweight spectra with large uncertainties in a given pixel in training the model for that pixel.

You may find it useful to work with just one or two wavelength pixels first, to experiment with different model architectures. Once you are satisfied with your model, you can train (i.e., fit for free parameters) the models for all pixels in parallel.

g) To ensure that your model is working properly, use it to predict the spectrum of one of the objects in the training set from its labels. Overplot the data and model spectra. They should look very similar.

Grad students only: *For each of the five labels ℓ_i, plot the gradient spectrum $\mathrm{d}f_\lambda/\mathrm{d}\ell_i$. This will let you identify which wavelengths are most sensitive to a particular label. For the gradient spectra of Si and Mg, mark the locations of strong known Si and Mg lines. Do the regions of the spectrum where the gradient is large correspond to known absorption lines?*

Some well-known lines at APOGEE wavelengths can be found here.

h) To test how well the model works, we'll now use it to fit for the labels of spectra in the cross-validation set. For each spectrum in the cross-validation set, use an optimizer (Python has many options) to find the point in label-space at which the spectrum predicted by the model best matches the observed spectrum (in a χ^2 sense, accounting for the uncertainty in the spectrum).

Now compare, for each of the five labels, the best-fit value obtained by the above procedure to the ASPCAP-derived value in the allStar catalog. Measure the bias and scatter for each label over the full cross-validation set. For a good model, these should be small; for example, a scatter of about 50 K in Teff and 0.05 dex in Fe/H should be achievable.

Grad students only: *Although your model should perform well in cross-validation in most cases, there are likely a few objects for which the best-fit labels differ substantially from those in the allStar catalog. Investigate these objects, and try to find out what has gone wrong. Did the optimizer get stuck in a local minimum? Is there something wrong with the spectrum or normalization? Are there flags in the catalog indicating the allStar labels might not be reliable? Can you improve your model based on these tests?*

i) For the ∼800 stars in your cross-validation set, plot a Kiel diagram (i.e. logg vs Teff). Color points by their Fe/H. Identify known features. Comment on the presence (or absence) of trends with Fe/H. The

paper by Holtzman et al. (2015) should give you a sense of what this is expected to look like.

Download and overplot a 6 Gyr-old MIST isochrone of solar metallicity. How good (or poor) is the agreement? Also plot an isochrone with Fe/H = -1. Does the Fe/H-trend in the isochrones agree with that found in your fitting?

j) Wrap your spectral model in MCMC. Then, use it to fit the provided **mystery spectrum**. As always, state your priors. Plot a corner plot of your constraints on the five labels.

Comment on the formal errors on your fit. Do they seem reasonable, too small, or too large? How do they compare to the typical errors of the ASPCAP-derived labels? How do they compare to the typical errors you inferred from the scatter in cross-validation? If the magnitude of the uncertainties is different from what you might expect, comment on factors that might explain this.

k) Use your model to make a movie showing how the spectrum changes with metallicity at fixed Teff and logg. For clarity, show only the region of the spectrum from 16000 to 16200 Angstrom. Vary Fe/H from -1 to 0.5 and fix the atmospheric parameters to reasonable values.

The matplotlib "animate" tool is useful for making movies. Unfortunately, it is not currently possible to save movies usign ffmpeg on Datahub. If you are working on your own computer, this should not be a problem. If you are using Datahub, a reasonable option is to save frames of the movie and then stitch them together.

Grad students only: *Make a movie showing how the same region of the spectrum changes as a star moves up the red giant branch. Fix [Fe/H]=0, and vary logg from 3.5 to 0.5, simultaneously varying Teff such that the star stays in a region of parameter space allowed by hydrostatic equilibrium.*

Comment on the similarities and differences of how the spectrum changes when the composition changes vs. when the star moves up the RGB at fixed composition. How can one tell the difference between a cool, low-logg star and a warmer, higher-logg star that is more metal-rich?

l) One complication not accounted for in your spectral model is the fact that many stars are in binary systems. If the angular separation between the two components of a binary is small, they will fall within the same spectroscopic fiber, and the observed spectrum will really be the sum of the spectra of two different stars. Discuss how this might affect your results. If it would lead to bias in the inferred labels, in which direction would you expect the bias to go? How might you correct for it?

The effects of binary are discussed in detail in El-Badry et al. 2017.

m) Finally, let's try a completely different way of measuring labels from spectra. So far, we've made a made a model to *predict the spectrum as a function of labels*, and then used that model to fit spectra in a traditional χ^2 sense. What if we could instead *predict labels as a function of spectra?* To do this, we'll use deep learning.

Train a neural network that takes in a normalized spectrum and predicts the same label vector that characterized your spectral model. You may find it useful to regularize the labels so they are of order unity. Experiment with different neural network architectures and hyper-parameters, carrying out on-the-fly validation to tune the network. Any neural network implementation is fine; the instructors have found PyTorch to work well.

Now use your neural network to predict the labels of stars in the cross validation set. Compare these to the true APSCAP labels, quantifying performance in terms of the bias and median error for each label. How does the network's performance compare to that of the model fitting you did in part (h)? Discuss the advantages and disadvantages of both strategies of measuring labels.