Lab 2: Modeling Stellar Spectra

Astro 128 (UC Berkeley)

Assigned: Mon Oct. 13, 2025
Checkpoints: Wed Oct. 22, 2025; Wed Oct. 29, 2025; Wed Nov. 5, 2025
Final Write Up Due: 11:59 PM Sun Nov. 9, 2025

Introduction

The goals of this lab are 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. Read it. Also check out Majewski et al. 2017 and Ahumada et al. 2020, which describe the APOGEE survey, and Holtzman et al. 2015, which describes the pipeline APOGEE uses to fit labels from spectra.

Technical Components

- Data munging and standardization
- Outlier rejection; removing bad data
- Linear models
- Data Driven Models
- Cross validation
- Nonlinear optimization, MCMC
- Basics of stellar evolution
- Neural networks

Broad Summary of Steps

The lab proceeds in the following order:

- Download stellar spectra and labels from SDSS/APOGEE using the specified quality cuts to trim the sample.
- Manipulate the spectra to get it in the desired format. This includes applying bitmasks to mitigate bad pixels and performing continuum normalization.
- Build a data-driven model that predicts flux at each wavelength. You will use a training set to determine the coefficients of your model and a validation set to protect against overfitting. This process is known as cross-validation.
- Use your data-driven model to predict the stellar labels for a larger set of stellar spectra. Several parts of this step are aimed at testing the robustness of your model and building your physical intuition of how spectra are influenced by stellar physics.
- Optional: Train a neural net to predict stellar labels, given a spectrum. Compare this to the parametric model approach you developed in the previous parts of the lab. This step can take a significant amount of time due to the computational expense of training a neural net. Budget your time accordingly.

Problem

1. To build a training set of stellar spectra and their corresponding properties (“labels”), we need to determine which stars we want to include in our training. The catalog that contains the labels derived by the ASPCAP pipeline for all spectra can be found in the “allStar” catalog. The entire catalog can be downloaded then sorted through or `astroquery.sdss.SDSS` can be used to query only the set of stars we are interested in for the training set.

   APOGEE spectra are sorted into different directories on the SDSS server based on their “field”, a few-digit string identifying their location in the sky. For this lab, we’ll download all the spectra with the following 4 (randomly chosen) fields: “M15”, “N6791”, “K2_C4_168-21”, and “060+00”. Additionally, due to data quality issues, not all labels have been derived for all stars. Discard all spectra for which:
   - There are no labels for T_{eff}, logg, [Fe/H], [Mg/Fe], [Si/Fe]
   - Low SNR spectra; SNR < 50, as reported in the allStar catalog)
   - Dwarf stars ($\log g > 4$ or $T_{\text{eff}} > 5,700\,\text{K}$); we’ll be focusing on giants in this lab.
   - stars with low metallicity ([Fe/H] < -1.

   After these cuts, you should be left with approximately 1880 stars in your training set (we have 1886). Do not spend too much time trying to get the same number of stars; a few missing or extra stars will not affect your model.
   Explain how our cut on $\log g$ effectively distinguishes between dwarfs and giants. Suppose you have a solar-mass star. Calculate the expected value of $\log g$ on the main sequence (when $R \sim 1R_\odot$), just before the helium flash (when $R \sim 100R_\odot$), and during core helium burning (when $R \sim 15R_\odot$).

   You should be left with 1855 stars. Visualize their distribution in label space using a corner plot.
2. Download the subset of the APOGEE spectra in the form of apStar files, following the instructions at this link or using the `astropy.io.fits` module. Note that APOGEE is part of SDSS, which has had many public data releases. The format in which the spectra are stored has changed several times between data releases, so make sure you are always accessing the most recent version of the spectra (DR17).

   Each spectrum should be in its own apStar “.fits” file, with a name like
   “apStar-dr17-2MXXXXXXXX+XXXXXXX.fits”. Here 2MXXXXXXXX+XXXXXXX is the 2MASS ID of the target. Each apStar file contains individual visit and coadded multi-visit spectra for one star. Helpfully, the coadded spectra have already been Doppler shifted to the barycentric frame. Explain 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 (flux vs. wavelength). What are the units of spectra? Explain what these units mean.
3. 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 (i.e., in the error array) in these pixels to a large value, so that they will not contribute significantly to the likelihood function in your fitting. The bitmask are a bit unintuitive but are described on in APOGEE data model. Read what each bit in the APOGEE_PIXMASK means for APOGEE spectra. Based on our experience, bits 0–7 and 12 are the most important; the others can probably be ignored.
4. Before we fit spectra, we need to “pseudo-continuum normalize” them, as described in Ness et al. 2015. Explain what this means and why it is useful. What is the difference between pseudo-continuum normalization and “true” continuum normalization?

   Developing a continuum normalization procedure from scratch is challenging: an iterative method is required to determine which wavelengths are insensitive to label changes (and thus good for fitting continuum). To find out more about this, read sections 2.3 and 5.3 of Ness et al. 2015.
   In this lab, we are making the problem a bit easier by providing you with a list of wavelengths (“con-

tinuum_pixels_apogee.npz") that do not contain any strong absorption lines. In other words, the flux value at these wavelengths should not depend much on the spectral labels of the star, but only on its absolute magnitude and distance. Write a function that uses the flux in these wavelength pixels to estimate the continuum over the full APOGEE wavelength range. Section 2.3 of Ness et al. 2015 should be useful. Because the APOGEE spectra are split over three chips, with gaps between the chips, your continuum-determination procedure will probably work better if you handle the three chips individually.

Normalize all your spectra and error arrays. Plot some example un-normalized spectrum, the derived pseudo-continuum, and the normalized spectrum. In order for the rest of your lab to be successful, it is critical that the normalization is robust and behaves as expected in all cases. We recommend testing it *thoroughly* before moving on to the next part of the lab. If your routine is robust, you should find that the normalized spectra of stars that have similar labels (according to ASPCAP) are very similar. Any artifacts (e.g. wiggles) introduced by your normalization procedure will reduce the performance of your spectral model.

Required checkpoint #1: your submission should contain (a) the corner plot of the stellar labels you made in problem (2), and (b) for star 2M21235315+1244123 submit a plot that has (i) its unnormalized spectrum, (ii) your fit to this star's continuum, and (iii) your normalized spectrum. Submit this via gradescope.
It should be a pdf, named Firstname_Lastname_lab2_cp1.pdf

5. 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.
6. **Note: Past students have indicated this is a particularly challenging 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: $T_{\rm eff}$, logg, [Fe/H], [Mg/Fe], [Si/Fe]. Following Ness et al., make your spectral model a 2nd-order polynomial in labels. Your final spectral model will consist of thousands of individual models—one for each wavelength pixel—stitched together. In addition to the model free parameters, fit an intrinsic scatter term, s_λ^2, at each wavelength. This term is defined such that the variance in the observed normalized flux values at wavelength λ is given by $s_\lambda^2 + \sigma_\lambda^2$, where σ_λ represents the uncertainty in the normalized flux.
   (a) Consider a single pixel of wavelength λ. Let $\mathbf{f}_\lambda$ be an array containing the normalized flux in that pixel for all stars in the training set. Show that *at a fixed value of* s_λ^2, the spectral model for the pixel can be described by a linear equation $\mathbf{X}\theta_\lambda = \mathbf{f}_\lambda$, where θ_λ is an array of model parameters for that pixel and $\mathbf{X}$ is a matrix that is the same for all pixels. What is $\mathbf{X}$? For a spectral model that is a 2nd order polynomial in labels, how many free parameters are in θ_λ (where $n = 5$ in your case)? Equation 8 of Ness et al. 2015 may provide some useful context here.
   (b) To find the best values of s_λ^2 and θ_λ for all pixels, consider a grid of s_λ^2 values for each pixel. Stepping though the grid, solve the linear equation for θ_λ and compute the likelihood for the training set for that value of s_λ^2. There should be a term in this likelihood that explicitly depends on s_λ. Choose the optimal θ_λ and s_λ^2 as the one that maximizes that likelihood. Ensure your grid of s_λ^2 is large and fine enough that you are actually finding a global maximum in the likelihood.
   (c) Repeat for all wavelength pixels. What is the total number of free parameters in your spectral model?
   (d) Now you have a trained spectral model consisting of your parameters $\{\theta_\lambda\}$. Write a function that takes a label vector for an arbitrary star and uses the model to predict the normalized spectrum.

   You may find it useful to:
   - rescale the labels such that they are all of order unity.
   - downweight spectra with large uncertainties in a given pixel in training the model for that pixel.

   If you are clever about how you formulate your model, you will be able to write down the model optimization as a linear algebra problem. So, even though you a training a complex model with many free parameters, the training should not take more than a few cpu minutes at most.
7. To ensure that your model is working properly, use it to predict the spectrum of some of the objects in the training set from its labels. Specifically, overplot the normalized spectrum of star 2M03533659+2512012 and the model spectrum predicted for its labels. Show the wavelength range from 16000 to 16100

Angstroms. The data and model spectra should look almost identical. If they don't, go back to step 6.

8. 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.

   Additionally, plot s^2_λ. Do wavelengths with larger-than-average intrinsic scatter correspond to known absorption lines?
9. 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 a nonlinear optimizer (Python has many options, we have found the "trf" or Trust Region Reflective method in scipy.optimize.curve_fit to be very robust) 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 for the validation set. That is, make plots of your best-fit labels vs the ASPCAP labels with a one-to-one line for reference, and show the residuals. 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 30 K in T_{eff} and 0.02 dex in [Fe/H] should be achievable. If you aren't happy with your model's performance, go back to step 6.
   Required checkpoint #2: submit a pdf containing (a) the plot you produced in part (7), and (b) the cross-validation plot produced in part (9). Submit this via gradescope.
   It should be a pdf, named Firstname_Lastname_lab2_cp2.pdf
10. 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 continuum normalization? Are there flags in the catalog indicating the allStar labels might not be reliable? Can you improve your model based on these tests?
11. For the ~900 stars in your cross-validation set, plot a Kiel diagram (i.e. logg vs T_{eff}). Color points by their Fe/H. You should use labels you obtained through fitting, not the ASPCAP labels. 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?
12. **Note: Past students have indicated this is a particularly challenging problem** Wrap your spectral model in MCMC using pymc. 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.

    Depending on how you have implemented your spectral model, it may or may not be easy to make it play nice with pymc. If it's not easy, a good workaround is to write the log-likelihood function yourself and enroll it manually. See this tutorial for a worked example of how this can be done.

    Comment on the formal parameter uncertainties 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.
    Required checkpoint #3: submit your (a) Kiel diagram from (11) and (b) the corner plot from your pymc fit in (12). Submit this via gradescope.
    It should be a pdf, named Firstname_Lastname_lab1_cp3.pdf
13. Use your model to a make a plot using color and offset spectra that shows how the spectrum changes

with metallicity at fixed $T_{\rm eff}$ 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.

14. Make a plot using color and offset spectra to show how the same region of the spectrum changes as a star ascends the red giant branch. Fix [Fe/H]=0, and vary logg from 3.5 to 0.5, simultaneously varying $T_{\rm eff}$ such that the star moves along an isochrone. 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?
15. **(optional for extra credit in AY128)**. 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. **Please note that training a neural net can take a fair amount of time (a) if you are entirely new to the process and (b) because the training of each model can take hours, and often, several instances of training / re-training are necessary. Please budget your lab time accordingly.**

    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; we have have found pytorch, TensorFlow/keras, and jax/equinox to work well. Once you find an architecture you are happy with, plot the training and validation loss as a function of training step. (The next lab uses PyTorch so we strongly suggest using that to begin getting familiar with the syntax.)

    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 (9)? Discuss the advantages and disadvantages of both strategies of measuring labels.