Fastai Lesson 6 Review

TWiML Fastai Meetup
Sat. 09 Mar 2019
Joseph Catanzarite

Topics

- **Kaggle Rossman Stores**
- **Regularization**
  - **Weight Decay**
  - **Dropout**
  - **BatchNorm**
  - Augmentation
- **CNNs**
- GradCAM
- Ethics

Agenda

- **Lesson 6 review 9:00 – 9:45**
- **Mini-presentation 9:45 – 10:15**
- **Open mic / chat 10:15 – 10:30**
- **Next up:**

  Sat. 3/16 Course1 v3: Lesson 7 review

  Sat. 3/23 Course 2 v3: Lesson 1 review

Rossmann Stores 1

3,000 drug stores in 7 European countries

Given a time series of data for 1115 German stores

Predict number of items sold in each store on each day over the subsequent 6 weeks

Root mean squared percent error

$$\text{RMSPE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{y_i}\right)^2}$$

- Typo: should be RMSFE: RMS *fractional* error
- By using log(y) instead of y as the 'label' and using the RMSE metric, we effectively get RMSFE
- Build the .pkl data archive by running this notebook from the Introduction to Machine Learning for Coders class

Fastai time series feature engineering

```
add_datepart(train,"Date", drop=False)
add_datepart(test,"Date", drop=False)
```

Adds new columns derived from "Date" timetag:

year, month, week of year, day of month, day of week, day of year

booleans start/end for, month, quarter, year

elapsed time since 1970

Rossmann Stores 2 Preprocesses

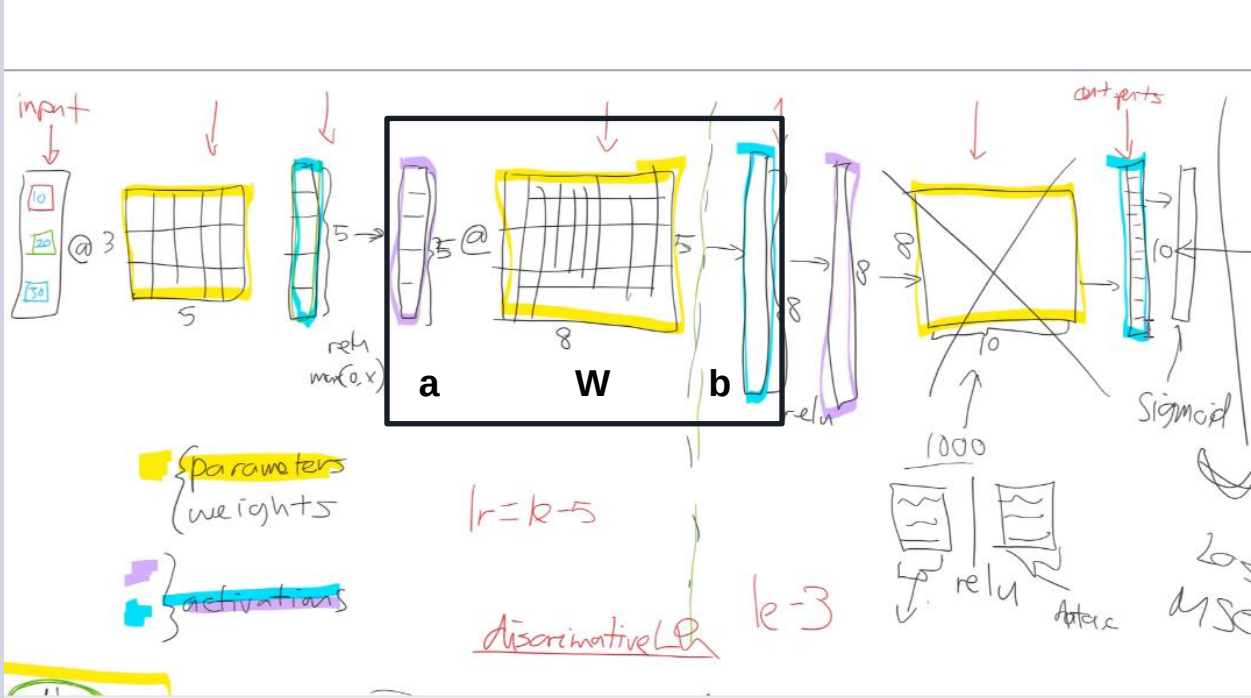
```
# define list of pre-processing step
procs = [FillMissing, Categorify,
Normalize]
# create databunch, passing in procs
data = (TabularList.from_df(df,
path=path, cat_names=cat_vars,
cont_names=cont_vars, procs=procs)
.split_by_idx(valid_idx)
.label_from_df(cols=dep_var,
label_cls=FloatList, log=True)
.databunch())
```

Run once on the training set and share metadata with the validation and test sets

1. `Preprocessor.Categorify(catvars, contvars, test=None)`
   - List the set of unique values for each category,
   - Map values to integer (>=0) numerical codes
   - Special code -1 for "missing" categorical value
   - Fastai adds 1 to each code so they can be used as lookup indices in embedding matrix
   - df.cat.codes lists the numerical codes
   - Numerical codes are metadata
2. `Preprocessor.FillMissing(catvars,contvars , test=None)`
   - For each contvar, automatically creates a column `contvar_na`, a boolean indicator for missing values.
   - Fills missing values with the median
   - Medians are metadata
3. `Preprocessor.Normalize`
   - For each contvar
     Subtract mean and divide by std
   - Means and standard deviations are metadata

Jeremy's notation for matrix multiplication: method in the madness?

- Purple: activations after applying RELU **a** [5 x 1]
- Yellow: weights matrix **W** [5 x 8]
- Cyan: raw activations **b** [8 x 1]
- We're trained to think of matrices multiplying column vectors from the left: $\mathbf{W}^T$ @ **a** = **b**
  Check: [8x5] @ [5x1] = [8x1]
- For NN, processing flows from left to right: so it's more natural to think **a** @ **W** = **b,** where
- The i^{th} element of **b** is the dot product of **a** with the i^{th} column of **W**



What is Regularization?

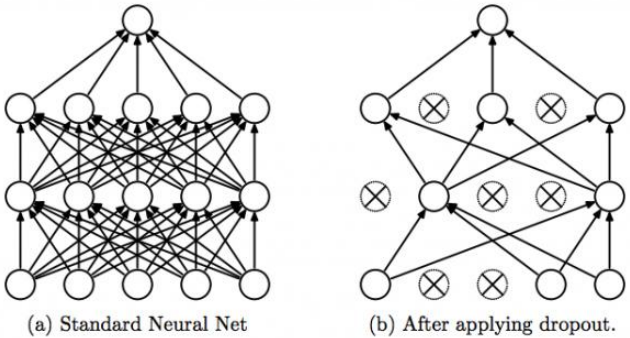
Conventional wisdom:
Avoid building complex models because they overfit!

Jeremy's reply (paraphrased):
Nonsense! Go ahead and build as complex a model as your compute resources can comfortably handle. Don't worry, you can control overfitting by regularization!

- Regularization methods are techniques that mess with model fitting process, making it more 'noisy'.
- Why? To prevent the model from overfitting.
- What is overfitting? The tendency of a model to 'learn' to memorize the training labels.
- The goal is to build a model that 'generalizes', i.e. doesn't overfit, and can perform well on data outside of the training set.
- Lesson 6 introduces 4 regularization methods
    - Dropout
    - Weight decay
    - Batch normalization
    - Data Augmentation

Dropout Regularization

- Adds ‘noise’ to the model
- For each minibatch, a different ‘perturbed’ version of the model generates the weight updates



(a) Standard Neural Net (b) After applying dropout.

Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

- For each minibatch: at each level **j**
- Randomly drop activations (i.e. set them to zero) with probability $\mathbf{p_j}$
- Scale the remaining activations by $\mathbf{1/p_j}$, so that the sum of the activations will be roughly constant
- Apply dropout during training, not during test.

Weight Decay Regularization

Perturb the model by adding a constraint that tries to keep the weights small

L2 regularization

- Add a multiple of the sum of the squares of the weights to the loss function.

$$L(x, y) \equiv \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2 + \lambda \sum_{i=1}^{n} \theta_i^2$$

- Constraining the weights to be small reduces the flexibility of the model to fit the data

Weight decay

- Adds the extra term to the weight update rule
- Does not change the loss function
- Accomplishes the same thing as L2 regularization

BatchNorm

- Regularizing effect is from use of momentum
- means and standard deviations in each minibatch are updated exponentially weighted moving averages
- Add Batchnorm layer for continuous variables
- Reduces variance of loss surface
- So you can use a higher learning rate

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

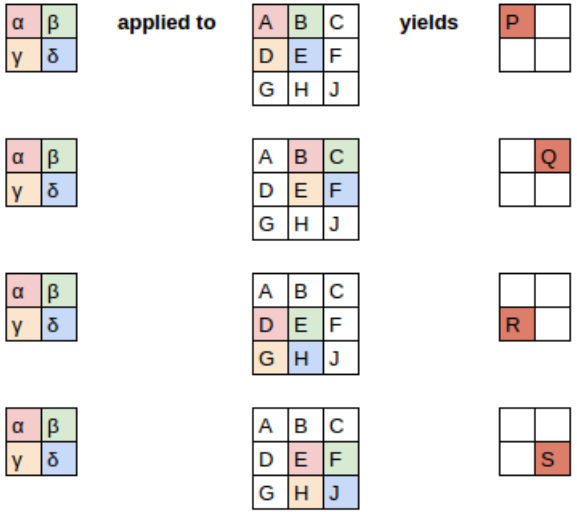
$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

- Epsilon is a constant added to stabilize the denominator in the variance: fastai uses 1e-5
- beta and gamma are learned by the model
- Scale and shift allows the model to improve match to data

Convolutional Neural Networks (CNNs)

- Kernels are filters that select coarse features in an image, such as edges
- Can use different kernel for each channel
- For an RGB image the kernel is 3-dimensional
- Padding
- Stride



$$\alpha A+\beta B+\gamma D+\delta E+b = P$$
$$\alpha B+\beta C+\gamma E+\delta F+b = Q$$
$$\alpha D+\beta E+\gamma G+\delta H+b = R$$
$$\alpha E+\beta F+\gamma H+\delta J+b = S$$

Applying a convolution kernel to an image