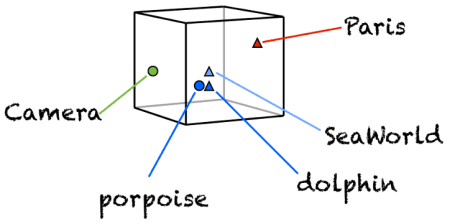
While the concept of embedding representation has been used in NLP for quite some time, the idea to represent categorical variables with embeddings appreared just recently

The image (from quora) quickly summarises the embedding concept. Words or categorical variables are represented by a point in n or in this case 3-dimensional space. Words which are similar are grouped together in the cube at a similar place.



In response to my post, I got the question of how to combine such embeddings with other variables to build a model with multiple variables.  
This is a good question and not straight-forward to achieve as the model structure inn Keras is slightly different from the typical sequential model.

As in the post before, let’s work with the nyc citi bike count data from Kaggle. It contains daily bicycle counts for major bridges in NYC.

In order to have a longer dataset, I use the bicycle count for all bridges as the dependent variable

#<https://www.kaggle.com/new-york-city/nyc-east-river-bicycle-crossings>  
df <- read.csv("data/nyc-east-river-bicycle-counts.csv")  
   
   
dflong <- data.table::melt(df[c("Date", "Brooklyn.Bridge", "Manhattan.Bridge", "Williamsburg.Bridge" ,"Queensboro.Bridge")], idvars="date")  
   
dflong$date <- as.Date(dflong$Date)  
dflong$weekday <- wday(dflong$date)  
   
dflong <- merge(dflong, df[, c("Date", "Precipitation", "Low.Temp..Â.F.")], by="Date")  
   
dflong$ScaledUsers <- scale(dflong$value)  
dflong$lowTemp <- scale(dflong[,"Low.Temp..Â.F."])  
   
dflong$rain <- ifelse(dflong$Precipitation != 0, 0,1)  
   
### important - replace string factors with numbers for Keras to work!  
dflong$Bridge <- factor(dflong$variable)  
levels(dflong$Bridge) <- 1:length(levels(dflong$Bridge))

Please note the code last lines. Keras works only with double and integer variables, hence we have to replace the Bridge-factor variable with indicies between 1 and 4.  
The goal of our play model is to predict the number of bicycle per day on a certain bridge dependent on the weekday, the bridge (“Brooklyn.Bridge”, “Manhattan.Bridge”, “Williamsburg.Bridge” ,”Queensboro.Bridge”), if it rains and the temperature. So overall we have 2 categorical variables, one binary and one continuous variable.

One simple way to use a deep net with this dataset is to “One-hot” encode the categorical variables, combine them in one dataframe.

require(keras)  
   
dfOneHot <- data.frame(to\_categorical(dflong$weekday, 8), to\_categorical(dflong$Bridge, 5), dflong$rain, dflong$lowTemp)  
   
model\_simple <- keras\_model\_sequential()  
   
model\_simple %>%  
 layer\_dense(input\_shape=(8+5+1+1), units=32, activation = "relu") %>%  
 layer\_dropout(0.4) %>%  
 layer\_dense(units=10, activation = "relu") %>%  
 layer\_dropout(0.2) %>%  
 layer\_dense(units=1)  
   
   
model\_simple %>% compile(loss = "mean\_squared\_error", optimizer = "sgd", metric="accuracy")   
   
model\_simple %>% fit(x = as.matrix(dfOneHot), y= as.matrix(dflong$ScaledUsers), epochs = 20, batch\_size = 5, validation\_split = 0.15)

Pretty straight-forward and the only points where people struggle is; setting the input correct (15) and to remember that we need to cast the dataframe to a matrix.  
So what about using embeddings instead of one-hot encodings? How can we combine two embeddings (weekday and bridge) with a binary (rain/no-rain) and continuous variable (temperature)?

First we define 3 input layers, one for every embedding and one the two variables. As both categorical variables are just a vector of lenght 1 the shape=1. For the last layer where we feed in the two other variables we need a shape of 2.

Next, we create the two embedding layer. The input dimension is the number of unique values +1, for the dimension we use last week’s rule of thumb.

Third, we concatenate the 3 layers and add the network’s structure.  
Finally, we use the keras\_model (not keras\_sequential\_model) to set create the model.

embedding\_size\_weekday=3  
embedding\_size\_bridge=2  
  
## 1  
inp1 <- layer\_input(shape = c(1), name = 'inp\_weekday')  
inp2 <- layer\_input(shape = c(1), name = 'inp\_bridge')  
inp3 <- layer\_input(shape = c(2), name = 'inp\_otherVars')  
  
## 2  
embedding\_out1 <- inp1 %>% layer\_embedding(input\_dim = 7+1, output\_dim = embedding\_size\_weekday, input\_length = 1, name="embedding\_weekday") %>% layer\_flatten()  
  
embedding\_out2 <- inp2 %>% layer\_embedding(input\_dim = 4+1, output\_dim = embedding\_size\_bridge, input\_length = 1, name="embedding\_bridge") %>% layer\_flatten()  
  
## 3  
combined\_model <- layer\_concatenate(c(embedding\_out1, embedding\_out2, inp3)) %>%  
 layer\_dense(units=32, activation = "relu") %>%  
 layer\_dropout(0.3) %>%  
 layer\_dense(units=10, activation = "relu") %>%  
 layer\_dropout(0.15) %>%  
 layer\_dense(units=1)  
  
## 4  
model <- keras::keras\_model(inputs = c(inp1, inp2, inp3), outputs = combined\_model)  
  
model %>% compile(loss = "mean\_squared\_error", optimizer = "sgd", metric="accuracy")   
  
summary(model)

## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## Layer (type) Output Shape Param # Connected to   
## ===========================================================================  
## inp\_weekday (InputLayer (None, 1) 0   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## inp\_bridge (InputLayer) (None, 1) 0   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## embedding\_weekday (Embe (None, 1, 3) 24 inp\_weekday[0][0]   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## embedding\_bridge (Embed (None, 1, 2) 10 inp\_bridge[0][0]   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## flatten\_35 (Flatten) (None, 3) 0 embedding\_weekday[0][0]   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## flatten\_36 (Flatten) (None, 2) 0 embedding\_bridge[0][0]   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## inp\_otherVars (InputLay (None, 2) 0   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## concatenate\_29 (Concate (None, 7) 0 flatten\_35[0][0]   
## flatten\_36[0][0]   
## inp\_otherVars[0][0]   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## dense\_126 (Dense) (None, 32) 256 concatenate\_29[0][0]   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## dropout\_33 (Dropout) (None, 32) 0 dense\_126[0][0]   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## dense\_127 (Dense) (None, 10) 330 dropout\_33[0][0]   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## dropout\_34 (Dropout) (None, 10) 0 dense\_127[0][0]   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## dense\_128 (Dense) (None, 1) 11 dropout\_34[0][0]   
## ===========================================================================  
## Total params: 631  
## Trainable params: 631  
## Non-trainable params: 0  
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The model summary shows that the input takes place at different times during training. The input of the “other” variables happens late in the process.

In order to train this model, we need to feed the data in a list structure. Each input layer gets their own list of elements.

### a list with an element for each individual input layer.  
inputVariables <- list(as.matrix(dflong$weekday), as.matrix(dflong$Bridge), as.matrix(dflong[,c("lowTemp","rain")]))  
   
model %>% fit(x = inputVariables,  
 y= as.matrix(dflong$ScaledUsers), epochs = 20, batch\_size = 2)

Comparing the two approaches, it is pretty clear that the one-hot encoding will stay the norm. It feels very artificial to represent categorical variables with embeddings in Keras. There is either room for a wrapper function to automatically create the input layer part or a redesign of layer\_concatenate function.