

## Lecture G2. Deep Learning 2

Sim, Min Kyu, Ph.D., [mksim@seoultech.ac.kr](mailto:mksim@seoultech.ac.kr)



서울과학기술대학교 데이터사이언스학과

- 1 I. About
- 2 II. Linear regression
- 3 III. Deep Forward Network

# I. About

## Diamonds dataset

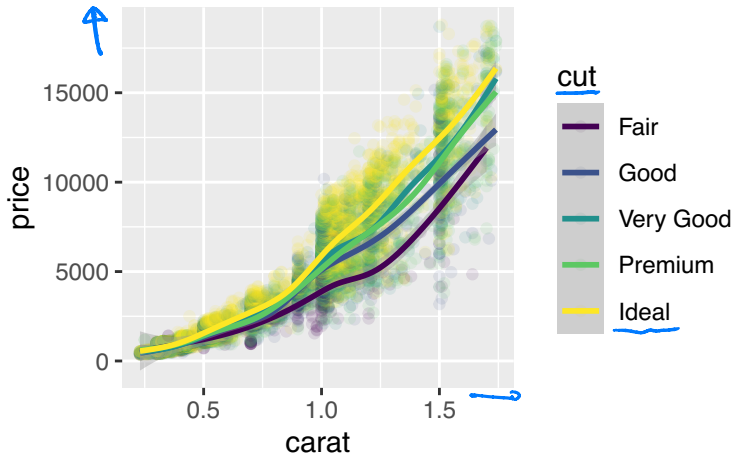
# Data Preparation

```
diamonds <- diamonds %>%  
  filter(color == "G", carat < 1.75) %>%  
  select(carat, cut, price)  
str(diamonds)  
  
## tibble [10,969 x 3] (S3: tbl_df/tbl/data.frame)  
## $ carat: num [1:10969] 0.23 0.23 0.28 0.31 0.31 0.24 0.7 0.78 0.74 0.75 ...  
## $ cut : Ord.factor w/ 5 levels "Fair"<"Good"<...: 3 5 5 3 4 4 5 3 5 4 ...  
## $ price: int [1:10969] 354 404 553 553 553 554 2757 2759 2760 2760 ...  
diamonds$cut %>% unique()  
  
## [1] Very Good Ideal Premium Good Fair  
## Levels: Fair < Good < Very Good < Premium < Ideal  
head(diamonds)  
  
## # A tibble: 5 x 3  
##   carat cut      price  
##   <dbl> <ord>   <int>  
## 1  0.23 Very Good    354  
## 2  0.23 Ideal       404  
## 3  0.28 Ideal       553  
## 4  0.31 Very Good    553  
## 5  0.31 Premium     553
```

price =  $f(\text{carat}, \text{cut})$

# Exploration

```
ggplot(diamonds, aes(x=carat, y=price, group=cut, color=cut)) +  
  geom_point(alpha=0.1) + geom_smooth()
```



# One-hot coding

```
library(mltools)
library(data.table)
diamonds$cut <- factor(diamonds$cut, ordered = FALSE)
diamonds <- diamonds %>% as.data.table() %>% one_hot()
head(diamonds)
```

	carat	cut_Fair	cut_Good	cut_Very Good	cut_Premium	cut_Ideal	price
## 1:	0.23	0	0	1	0	0	354
## 2:	0.23	0	0	0	0	1	404
## 3:	0.28	0	0	0	0	1	553
## 4:	0.31	0	0	1	0	0	553
## 5:	0.31	0	0	0	1	0	553
## 6:	0.24	0	0	0	1	0	554

$$Y = f(x)$$



## II. Linear regression

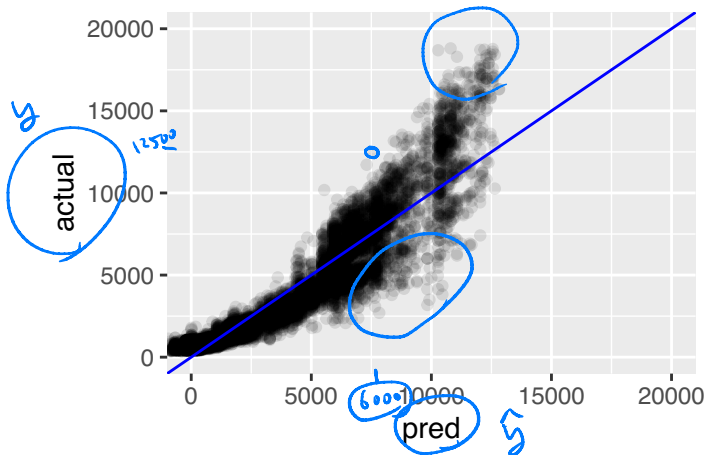


```
lm_fit <- lm(price ~ carat + cut_Fair + cut_Good + `cut_Very Good` + cut_Premium, data=diamonds)
summary(lm_fit)
```

```
##
## Call:
## lm(formula = price ~ carat + cut_Fair + cut_Good + `cut_Very Good` +
##      cut_Premium, data = diamonds)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -6654   -730     46    625   8409
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -2427.4      29.7    -81.8  <2e-16 ***
## carat          8761.3      33.3    263.2  <2e-16 ***
## cut_Fair       -1941.2      82.5    -23.5  <2e-16 ***
## cut_Good       -881.8      50.1    -17.6  <2e-16 ***
## `cut_Very Good` -432.8      34.1    -12.7  <2e-16 ***
## cut_Premium    -430.7      31.9    -13.5  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1330 on 10963 degrees of freedom
## Multiple R-squared: 0.864, Adjusted R-squared: 0.864
## F-statistic: 1.39e+04 on 5 and 10963 DF, p-value: <2e-16
```

## Fitted vs Actual

```
lm_fitted <- data.frame(pred=lm_fit$fitted.values, actual=diamonds$price)
ggplot(lm_fitted, aes(x=pred, y=actual)) +
  geom_point(alpha=0.1) + geom_abline(slope = 1, intercept = 0, color = "blue", size = 0.5) +
  coord_cartesian(xlim = c(0, 20000), ylim = c(0, 20000))
```

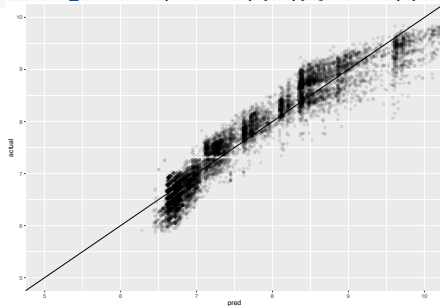


## Side note

```
lm_fit_log <- lm(  
  log(price) ~  
  carat + cut_Fair + cut_Good + `cut_Very Good` +  
  data=diamonds)  
summary(lm_fit_log)$r.squared
```

```
## [1] 0.9184
```

```
lm_fitted_log <- data.frame(  
  pred=lm_fit_log$fitted.values,  
  actual=log(diamonds$price))  
ggplot(lm_fitted_log, aes(x=pred, y=actual)) +  
  geom_point(alpha=0.1) +  
  geom_abline(slope = 1, intercept = 0) +  
  coord_cartesian(xlim = c(5,10), ylim = c(5,10))
```



## III. Deep Forward Network

# Keras and Tensorflow


- Webpage <https://tensorflow.rstudio.com/> ✓
- Installation <https://tensorflow.rstudio.com/installation/> ✓

## 1 Anaconda installation

- <https://www.anaconda.com/products/individual> ✓

## 2 Install package tensorflow and keras

```
install.packages("tensorflow")  
library(tensorflow)  
install_tensorflow()  
install.packages("keras")
```



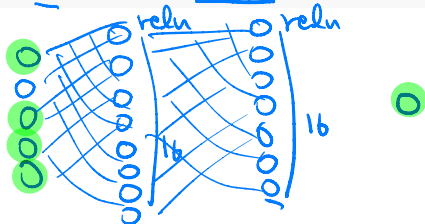
## 3 Confirm installation

```
library(tensorflow)  
tf$constant("Hello Tensorflow")
```



# 1. Construct a network

```
library(keras)
X <- diamonds %>% select(-cut_Ideal, -price) %>% as.matrix() ✓
Y_actual <- diamonds %>% select(price) %>% as.matrix() ✓
dim(X)
## [1] 10969 5
dim(Y_actual)
## [1] 10969 1
dense_fit <- keras_model_sequential() %>%
  layer_dense(units = 16, input_shape = c(5), activation = "relu") %>%
  layer_dense(units = 16, activation = "relu") %>%
  layer_dense(1, activation = "linear")
```



```
dense_fit %>% summary()
```

```
## Model: "sequential"
```

```
##
```

Layer (type)	Output Shape	Param #
--------------	--------------	---------

=====		
-------	--	--

dense (Dense)	(None, 16)	96
---------------	------------	----

```
##
```

dense_1 (Dense)	(None, 16)	272
-----------------	------------	-----

```
##
```

dense_2 (Dense)	(None, 1)	17
-----------------	-----------	----

=====		
-------	--	--

```
## Total params: 385
```

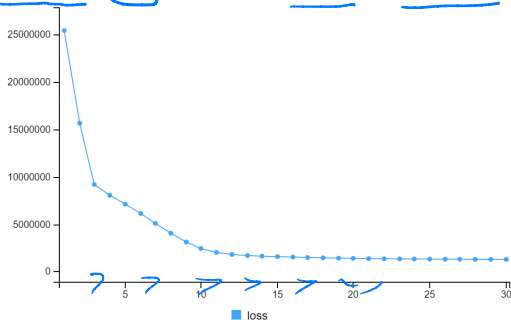
```
## Trainable params: 385
```

```
## Non-trainable params: 0
```

```
##
```

## 2. Compile and Fit

```
dense_fit %>% compile(loss = "mse", optimizer = "adam")  
dense_fit %>% fit(X, Y_actual, epochs=30, batch_size=16)
```





### 3. Predict

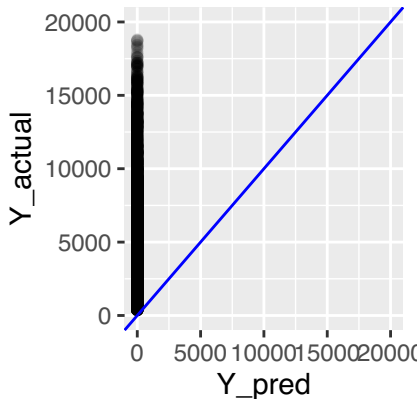
```
Y_pred <- dense_fit %>% predict(X)
results <- data.frame(Y_pred, Y_actual)
head(results)
```

```
##      Y_pred price
## 1 0.159352   354
## 2 0.007258   404
## 3 0.008835   553
## 4 0.171762   553
## 5 0.003778   553
## 6 0.004952   554
```

```
r_squared <- cor(Y_pred, Y_actual)^2
r_squared
```

```
##           price
## [1,] 0.002585
```

```
ggplot(results, aes(x=Y_pred, y=Y_actual)) +  
  geom_point(alpha=0.1) +  
  geom_abline(slope = 1, intercept = 0, color = "blue", size = 0.5) +  
  coord_cartesian(xlim = c (0, 20000), ylim = c (0, 20000))
```



"It's not that I'm so smart, it's just that I stay with problems longer. - A. Einstein"