Human Protein Recognition and Predicting Home Values

Mike Crumley

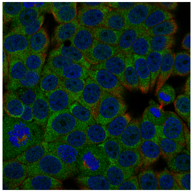
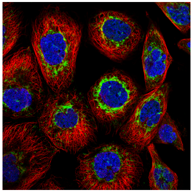
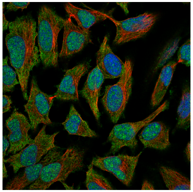
Project Overviews

1. Human Protein Recognition
   - Recognize presence/non-presence of 28 protein structures from microscopic cell images
   - Multi-class, multi-label classification
   - Software: Keras (R, on top of TensorFlow)
   - Metric: Macro F_1 score

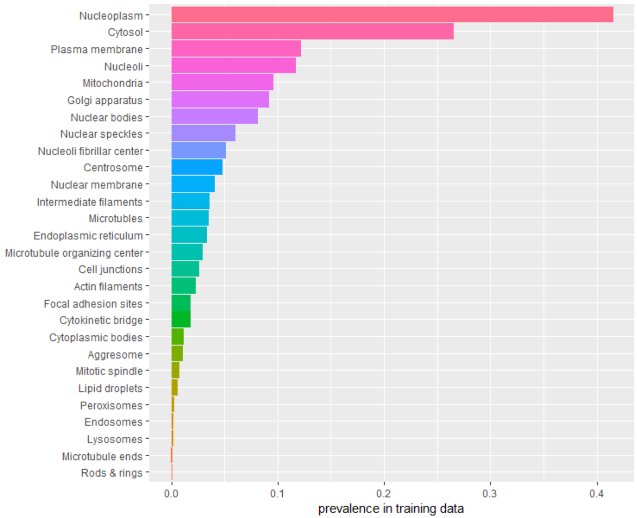
2. Predicting House Prices
   - Predict the sale price of a home using 78 predictors
   - Regression
   - Software: R (various packages)
   - Metric: root mean square logarithmic error

Protein Recognition: Data Exploration

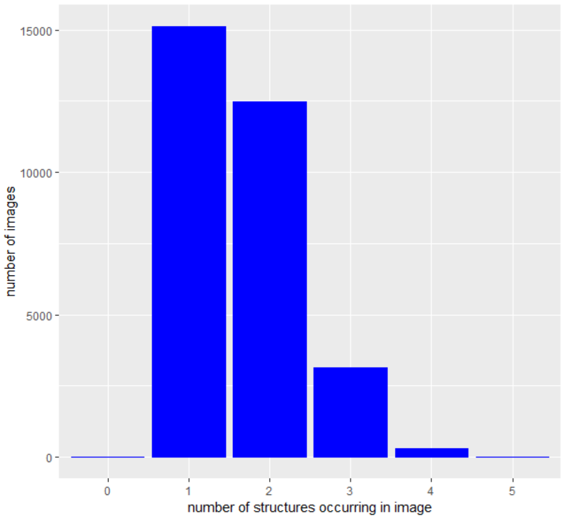
- Training data: $31,072$ RGB images (512×512)
- Test data: $11,702$ images

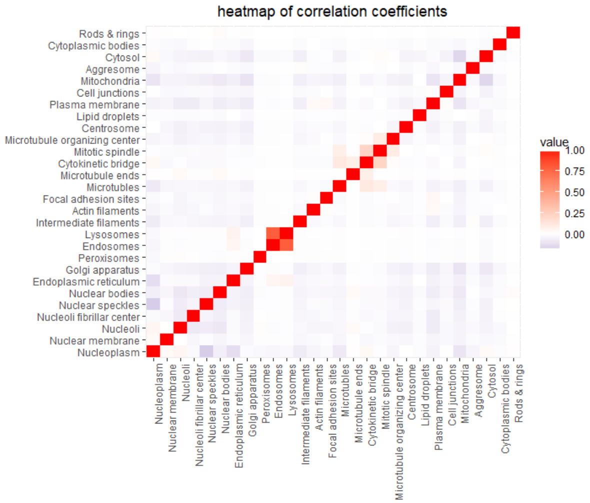
Protein Recognition: Data Exploration



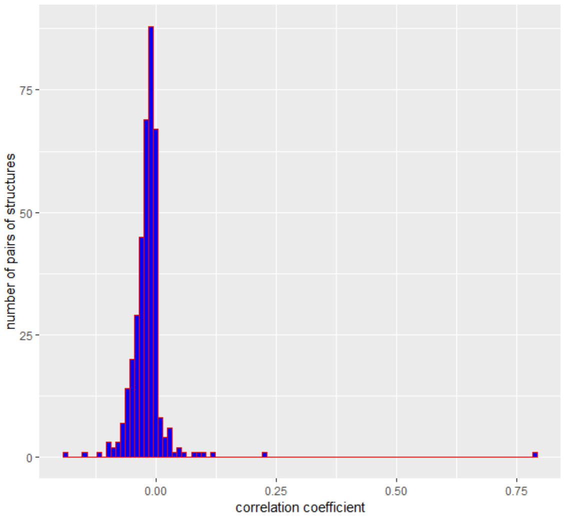
Protein Recognition: Data Exploration



Protein Recognition: Data Exploration



Protein Recognition: Data Exploration



Protein Recognition: Metric

For a single protein structure:

$$F_1 = \left(\frac{\text{precision}^{-1} + \text{recall}^{-1}}{2} \right)^{-1}$$

$$\begin{aligned} \text{precision} &= \text{fraction of images predicted to contain the structure} \\ &\quad \text{which actually contained it} \\ \text{recall} &= \text{fraction of images containing the structure} \\ &\quad \text{which were predicted to be so} \end{aligned}$$

For the entire multi-class problem:

$$\overline{F_1} = \left(\frac{\overline{\text{precision}}^{-1} + \overline{\text{recall}}^{-1}}{2} \right)^{-1}$$

Protein Recognition: Metric

Important notes about F_1:

- Penalizes false negatives more than false positives
- A baseline classifier for F_1 predicts *all* structures to be present in *all images*. A baseline macro F_1 score for this project is

  $$F_{1,B} = .1103$$

  Baseline scores for individual structures depend heavily on their prevalence; they range from .0007 to .586.
- It is non-differentiable, and can't be used as a loss function (though something close to it can be).

Protein Recognition: Convolutional Networks

Convolutional networks are the thing to do for image recognition.

An example of a convolutional neural network in Keras:

```r
model <- keras_model_sequential() %>%
  layer_conv_2d(filters = 16, kernel_size = c(10, 10), activation = "relu",
                input_shape = c(512, 512, 3)) %>%
  layer_max_pooling_2d(pool_size = c(5, 5)) %>%
  layer_conv_2d(filters = 32, kernel_size = c(10, 10), activation = "relu") %>%
  layer_max_pooling_2d(pool_size = c(4, 4)) %>%
  layer_conv_2d(filters = 64, kernel_size = c(3, 3), activation = "relu") %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 128, kernel_size = c(3, 3), activation = "relu") %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_flatten() %>%
  layer_dense(units = 256, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")
```

Protein Recognition: Convolutional Networks

```
## Model
## ___________________________________________________________________________
## Layer (type)                     Output Shape                  Param #
## ===========================================================================
## conv2d_1 (Conv2D)                (None, 503, 503, 16)          4816
## ___________________________________________________________________________
## max_pooling2d_1 (MaxPooling2D)   (None, 100, 100, 16)          0
## ___________________________________________________________________________
## conv2d_2 (Conv2D)                (None, 91, 91, 32)            51232
## ___________________________________________________________________________
## max_pooling2d_2 (MaxPooling2D)   (None, 22, 22, 32)            0
## ___________________________________________________________________________
## conv2d_3 (Conv2D)                (None, 20, 20, 64)            18496
## ___________________________________________________________________________
## max_pooling2d_3 (MaxPooling2D)   (None, 10, 10, 64)            0
## ___________________________________________________________________________
## conv2d_4 (Conv2D)                (None, 8, 8, 128)             73856
## ___________________________________________________________________________
## max_pooling2d_4 (MaxPooling2D)   (None, 4, 4, 128)             0
## ___________________________________________________________________________
## flatten_1 (Flatten)              (None, 2048)                  0
## ___________________________________________________________________________
## dense_1 (Dense)                  (None, 256)                   524544
## ___________________________________________________________________________
## dense_2 (Dense)                  (None, 1)                     257
## ===========================================================================
## Total params: 673,201
## Trainable params: 673,201
## Non-trainable params: 0
## ___________________________________________________________________________
```

Protein Recognition: Special Considerations

- Highly standardized laboratory images
- Extreme rarity of some structures
- Poor correlation between structures
- Typical loss functions (binary crossentropy, categorical cross entropy) can be very poorly aligned with the F_1 metric, especially for rare structures
- Validation/testing will suffer from rarity. Beware of over-fitting to validation.

Protein Recognition: Two Approaches

1. Training 28 individual binary classification models and ensembling the results. Advantages:
   - Can artificially balance individual classes, allowing networks to see more examples of rare structures
   - Low correlation between structures partially justifies this
   - Can tailor network architecture/optimizer for each individual structure
   - Possibly lower network complexity (my GPU is not impressive)
   - Learn more
2. Training a single model to predict all 28 protein structures simultaneously. Advantages:
   - Easier and quicker
   - Can use a custom loss to train directly for macro F_1
   - Low-mid level representations learned by a network may be useful for many structures

Protein Recognition: 28 Separate Models

Possible tools to combat rarity of structures:

- Custom loss function (penalize false negatives more than false positives)
- Data augmentation (geometric transformations to generate more training images)
- Artificial class balancing (allow network to see a higher proportion of structure-containing images)

Protein Recognition: 28 Separate Models

```r
# custom loss/metrics -------------------------------

f1.loss = function(y_true, y_pred)
{
  tp = k_mean(y_true*y_pred)
  fp = k_mean((1-y_true)*y_pred)
  fn = k_mean(y_true*(1-y_pred))
  f1 = 2*tp/(2*tp+fn+fp+k_epsilon())
  return((1-f1))
}

F1.macro.fcn.hard = function(y_true, y_pred)
{
  y_pred_hard = tf$floor(2*y_pred/(1+k_epsilon()))
  tp = tf$diag_part(k_dot(k_transpose(y_true),y_pred_hard))
  fn = tf$diag_part(k_dot(k_transpose(y_true),1-y_pred_hard))
  fp = tf$diag_part(k_dot(k_transpose(1-y_true),y_pred_hard))
  prec = tp/(tp+fp+k_epsilon())
  rec = tp/(tp+fn+k_epsilon())
  F1 =  2*(k_mean(prec)*k_mean(rec))/(k_mean(prec)+k_mean(rec)+k_epsilon())
  return(F1)
}

F1_macro_metric_hard <- custom_metric("Hard_F1", F1.macro.fcn.hard)
```

Protein Recognition: 28 Separate Models

```
# image data generators for training ------------------

train_datagen_rare = image_data_generator(
  featurewise_center = FALSE,
  samplewise_center = FALSE,
  featurewise_std_normalization = FALSE,
  samplewise_std_normalization = FALSE,
  zca_whitening = FALSE,
  zca_epsilon = 1e-06,
  rotation_range = 45,
  width_shift_range = .2,
  height_shift_range = .2,
  brightness_range = NULL,
  shear_range = 0,
  zoom_range = 0,
  channel_shift_range = 0,
  horizontal_flip = TRUE,
  vertical_flip = TRUE,
  rescale = 1/255,
)

train_datagen_notrare = image_data_generator(
  rescale = 1/255,
  rotation_range = 25,
  width_shift_range = .1,
  height_shift_range = .1,
  shear_range = 0,
  zoom_range = 0,
  horizontal_flip = TRUE,
  vertical_flip = TRUE,
)
```

Protein Recognition: 28 Separate Models

```r
# image data generators for validation -----------------

validate_datagen_rare = image_data_generator(
  featurewise_center = FALSE,
  samplewise_center = FALSE,
  featurewise_std_normalization = FALSE,
  samplewise_std_normalization = FALSE,
  zca_whitening = FALSE,
  zca_epsilon = 1e-06,
  rotation_range = 10,
  width_shift_range = .05,
  height_shift_range = .05,
  brightness_range = NULL,
  shear_range = 0,
  zoom_range = 0,
  channel_shift_range = 0,
  horizontal_flip = TRUE,
  vertical_flip = TRUE,
  rescale = 1/255
)

validate_datagen_notrare = image_data_generator(
  rescale = 1/255,
  horizontal_flip = TRUE,
  vertical_flip = TRUE
)
```

Protein Recognition: 28 Separate Models

```
# image data generators for testing -----------------

test_datagen_rare = image_data_generator(
  horizontal_flip = TRUE,
  vertical_flip = TRUE,
  rescale = 1/255
)

test_datagen_notrare = image_data_generator(
  rescale = 1/255
)
```

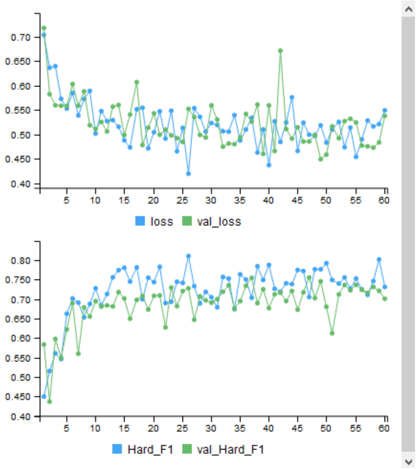
Protein Recognition: 28 Separate Models

```r
# custom training generator for artificial class balancing -------

biased_rgb_train_generator = function()
{
  function()
  {
    rand.numbers = runif(batch_size)
    present_indices = which(rand.numbers < training.bias[structure])
    notpresent_indices = setdiff(1:batch_size, present_indices)
    training_array = array(dim = c(batch_size, resolution, 3))
    label_array = array(dim = c(batch_size), 0)
    for (i in present_indices)
    {
      training_array[i, , , ] = generator_next(train_generator_present)
    }
    for (i in notpresent_indices)
    {
      training_array[i, , , ] = generator_next(train_generator_notpresent)
    }
    label_array[present_indices] = 1
    return(list(training_array, label_array))
  }
}
```

Protein Recognition: 28 Separate Models

```r
# train a bunch of models on 28 structures -------------

train.existing.models = function(
  model.folder.name,
  structures = protein.names,
  resolution = c(512,512),
  training.bias = default.training.bias,
  batch_size = 20,
  steps_per_epoch = 20,
  epochs = 20,
  save.history.name = "history.Rda",
  validation_steps = 100)
{
  ...
}
```

Protein Recognition: 28 Separate Models



Protein Recognition: 28 Separate Models

```r
# score a trained model on test data, summarize, find optimal thresholds ------

F1.score = function(
    model.folder.name,
    structures = protein.names,
    num_present_to_score = 1000,
    num_notpresent_to_score = 1000,
    resolution = c(512,512),
    save.name = "F1_summary.Rda",
    adjust.thresholds = TRUE)
{
  ...
}
```

Protein Recognition: 28 Separate Models

```
# find best models for each structure -------

find.best.ensemble(
  model.folder.names,
  adjust.thresholds = TRUE )
{
  ...
}
```

Protein Recognition: 28 Separate Models

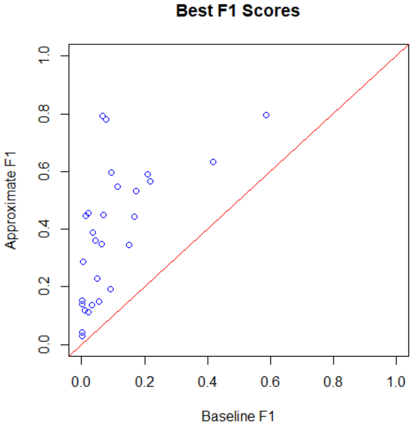
```r
# retrain best models on entire training set --------

train.existing.models.full = function(
  model.folder.name,
  structures = protein.names,
  resolution = c(512,512),
  training.bias = default.training.bias,
  batch_size = 20,
  steps_per_epoch = 20,
  epochs = 80,
  save.history.name = "history.Rda")
{
  ...
}
```

Protein Recognition: 28 Separate Models

```r
# make a submission file to kaggle -------

kaggle.submission = function(
  model.folder.name,
  resolution = c(512,512),
  adjust.thresholds = TRUE,
  save.name = "kaggle.submission.csv")
{
  ...
}
```

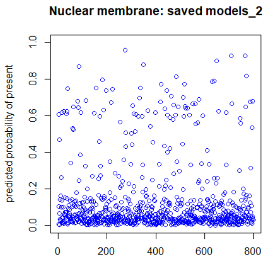
Protein Recognition: Results

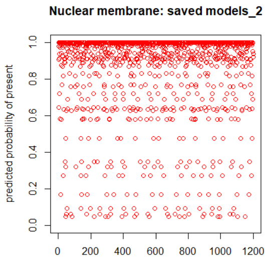
Kaggle score: .24 Macro F_1.



Protein Recognition: Results

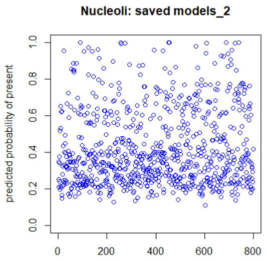
Good:

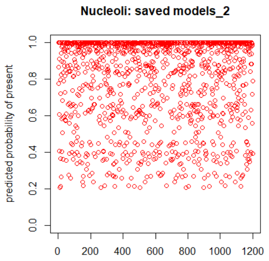




Protein Recognition: Results

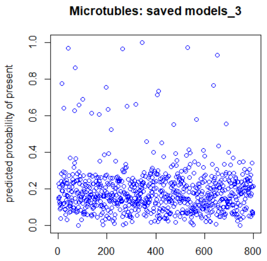
Good:

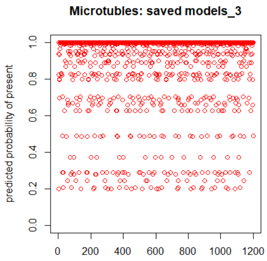




Protein Recognition: Results

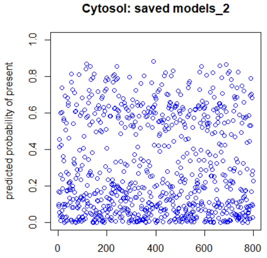
Good:

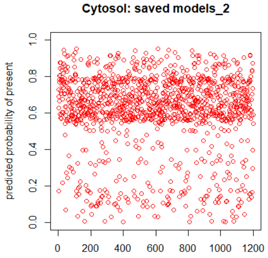




Protein Recognition: Results

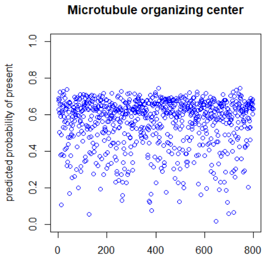
Good:

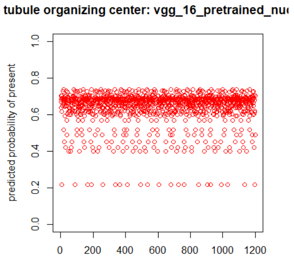




Protein Recognition: Results

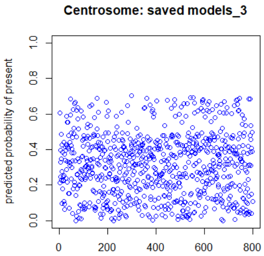
Not good:

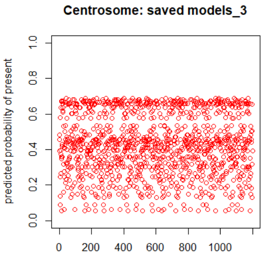




Protein Recognition: Results

Not good:





House Prices: Overview

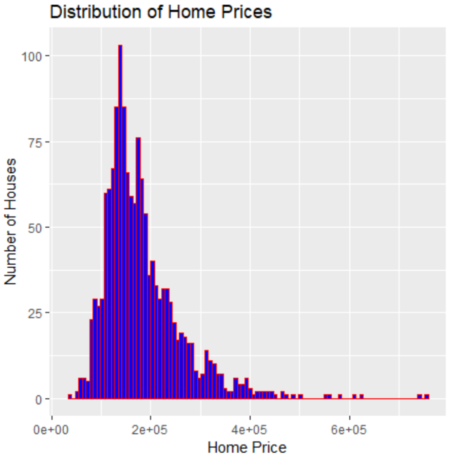
- To predict: Sale price of a home
- 78 predictors (square footage, bedrooms, neighborhood, etc.)
- Training data: 1460 homes (price known)
- Test data: 1459 homes (price unknown)

House Prices: Metric

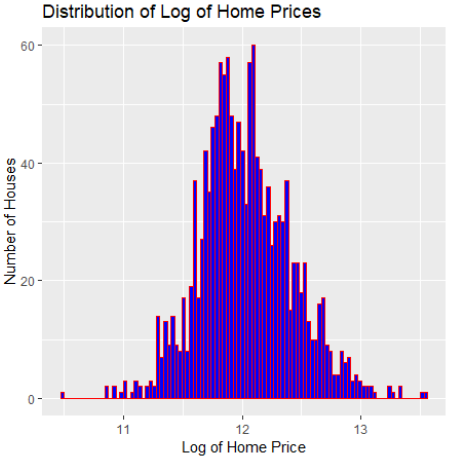
Root mean square logarithmic error:

$$RMSLE^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\log(1+\hat{y}_i) - \log(1+y_i)\right)^2$$

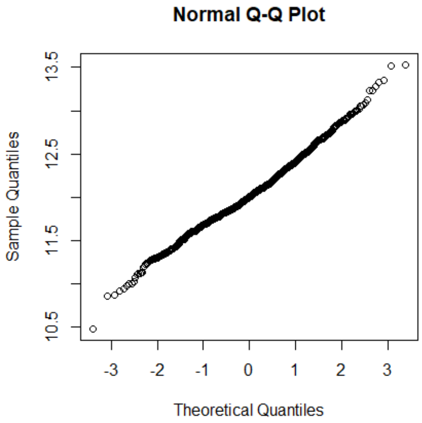
House Prices: Data Exploration



House Prices: Data Exploration



House Prices: Data Exploration



House Prices: Special Considerations

- Incomplete, redundant, and poorly formatted data
- Sale price is log-normally distributed
- Some predictors will be more informative than others
- Lots of models could possibly work here

House Prices: Data Cleanup

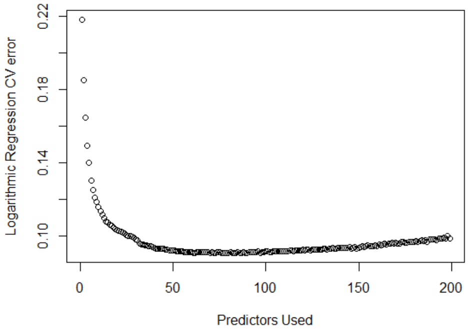
Steps taken to make the data usable:

- Data orthogonalization
- Level condensing
- Rank-deficiency correction
- Outlier removal

House Prices: Results

Learning Method	CV method	CV RMSLE	Test RMSLE	Notes
ordinary least squares	K = 10, B = 20	0.0988	0.13	all predictors used
least squares with subset selection	K= 10, B = 20	0.0908	0.1264	Forward subset selection identified 199 possible predictor subsets; CV was performed on each; lowest CV achieved with 83 predictors; this was submitted for test RMSLE
principal components regression	LOOCV	0.0989	0.1294	lowest CV achieved with 195 out of 199 principal components; this was submitted for test RMSLE
ridge regression	K = 10, B = 1	0.098	0.12419	Examined 150 values of λ from 10^{10} to 10^{-5}; lowest CV achieved at $\lambda = .01$; this was submitted for test RMSLE
LASSO	K = 10, B = 1	0.09587	0.12641	Lowest CV achieved at $\lambda = .0001913724$; this was submited for test RMSLE
regression tree	K = 10, B = 5	0.203		Didn't bother submitting, CV error too high
regression bag	OOB	0.1261	0.1528	500 trees grown (OOB error optimized long before that)
random forest	OOB	0.1234	0.1506	9 predictors at a time considered, 500 trees grown (OOB error optimized long before that)

House Prices: A Cool Graph



Thank you!