

AI for Medical Image Classification -Illustrated With Chest X-Rays

August 9-14, 2020 at Academia Sinica

Model Development Workflow (II)

陳素雲

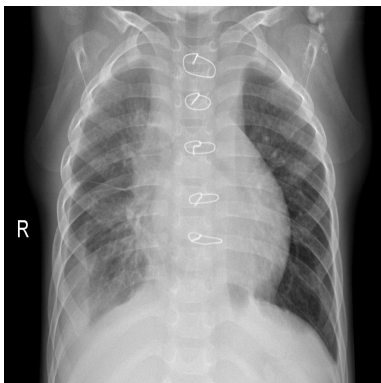
August 13, 2020

- 1 Data Problem
- 2 Statistical Models
- 3 Fitting Models (need loss criterion or divergence)
- 4 Model Selection
- 5 Develop One's Own Methodologies

Outline

- 1 Data Problem
- 2 Statistical Models
- 3 Fitting Models (need loss criterion or divergence)
- 4 Model Selection
- 5 Develop One's Own Methodologies

Training data: x-ray images with disease labels



224×224

with label (normal or pneumonia)

Task: diagnosis (label prediction) for test images

輸入的例子

$x =$



某一個人的肺部影像

$$f_l \circ f_{l-1} \circ \cdots \circ f_1(x)$$



神經網路把 x 轉換為 y

輸出的例子

$y = 0.145\dots$

左方肺部影像的人罹患肺炎的機率

model $f_l \circ f_{l-1} \circ \cdots \circ f_1$: dimension reduction (multilinear PCA, pooling, TensorProjection layer), convolution operators, (batch) normalization (pixel-wise standardization), dropout (regularization not dimension reduction), etc.

Outline

- 1 Data Problem
- 2 Statistical Models**
- 3 Fitting Models (need loss criterion or divergence)
- 4 Model Selection
- 5 Develop One's Own Methodologies

popular machine learning algorithms

logistic regression, linear SVM, kernel SVM, CNN in this short course

&

classical statistical viewpoints

modeling, shallow models, deep models

dimension reduction ↓

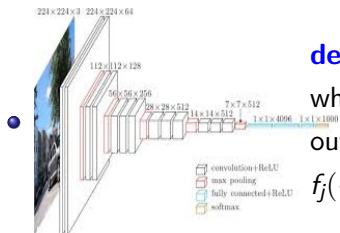
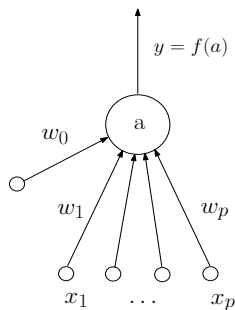
map to high dimensional feature space ↑

f_θ : modeling classifier

linear classifiers: logistic reg, SVM

- decision function: $w^\top \text{vec}(x) + b$

parameters w and b



deep NN: $y = f_\ell \circ f_{\ell-1} \cdots \circ f_1(x)$,
 where x is an input image and y is the output (disease probability prediction)

$f_j(\cdot | \theta_j)$; **parameters $\theta_1, \dots, \theta_\ell$**

Linear classifiers → nonlinear classifiers

- Linear: logistic regression, linear SVM
- Nonlinear: kernel SVM (shallow, one resolution)¹
 - multiple kernel SVM (shallow model, a couple of resolutions)
 - deep CNN (quite some different resolutions)

¹nonlinear in the original input space, but linear in the kernel induced feature space

Simple model vs. complex model

dataset size n and model complexity (number of parameters p)

- **large- p -small- n** (a.k.a. HDLSS) problems
- To infer p (large) parameters using only n (small) data points: it is a very challenging task and we often impose some conditions (e.g. dropout regularization, ℓ_1 constraint for sparseness).
- How reliable is the inference? Does $\hat{y} = 0.99$ mean that we can be very confident that the associated image input x shows pneumonia? The answer is "No", unless we can provide some statistical justification.²

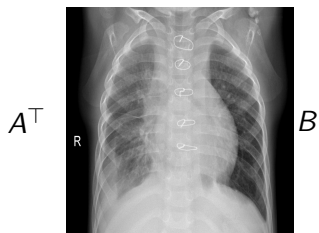
²large sample asymptotics (?), bootstrap, influence function (functional derivative wrt data dist)

Dimension reduction vs. mapping to high dimensional feature space (such as kernel map)

- Statistical inference in lower-dimension is often more efficient
- Higher dimension allows more flexible models

There is a trade-off between estimation efficiency and model bias.

Multilinear PCA for dimension reduction



$p_1 \times p_2$ reduced to $q_1 \times q_2$
(224×224 to 5×5)

logistic regression on $\text{vec}([\cdot]_{5 \times 5})$

CNN on 224×224 images

- stability from run to run, wrt data size, epochs, tuning parameters, data perturbation, etc.

Outline

- 1 Data Problem
- 2 Statistical Models
- 3 Fitting Models (need loss criterion or divergence)**
- 4 Model Selection
- 5 Develop One's Own Methodologies

Loss function (fitting criterion)

logistic regression, MLE, minimum KL-divergence estimation, binary cross-entropy loss,

- training data: $\{x_i, t_i\}_{i=1}^n$
where x_i is the i^{th} instance with label $t_i \in \{0, 1\}$
- model $y = f_{\theta}(x)$ indexed by the parameter θ
- loss function: $D(f^{\text{data}}, f_{\theta}^{\text{model}})$ (a divergence between data distribution and model distribution)

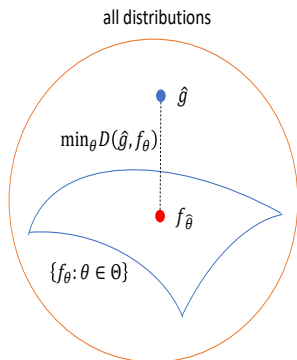
Kullback-Leibler divergence: MLE, cross-entropy loss

- **γ -divergence (day 5):** handling outliers and mislabels

Slide from Day 1 by Prof. Hung

A Quick Summary for Classification

- Random vector (X, Y)
- Bayes classifier: $\hat{Y} = \operatorname{argmax}_j \pi_j(X)$
 - Target: $\pi_j(X) = P(Y = j|X)$
- Model: $\pi_j(X) \stackrel{m}{=} \pi_j(X; \theta) \rightarrow$ **model distribution** f_θ
- Data: $\{(X_i, Y_i)\}_{i=1}^n \rightarrow$ **data distribution** \hat{g}
- Estimating θ via a proper D
 - $\min_{\theta} D(\hat{g}, f_\theta) + \lambda J(\theta) \rightarrow \hat{\theta}$ and, hence, $\pi_j(X; \hat{\theta})$



Outline

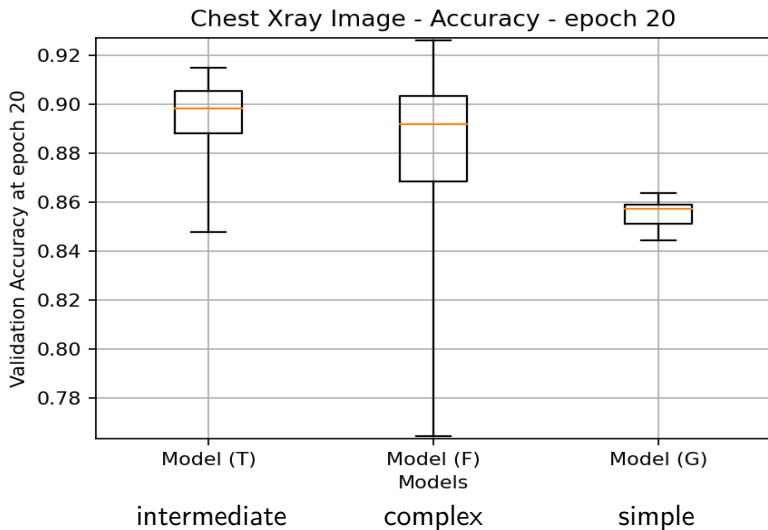
- 1 Data Problem
- 2 Statistical Models
- 3 Fitting Models (need loss criterion or divergence)
- 4 Model Selection**
- 5 Develop One's Own Methodologies

Model selection

Model selection has been an active field in statistical science.

- training (including validation) data goodness of fit
+ model complexity
- goodness of fit: evaluation metric
- model choices: $\mathcal{M} = \{M_1, \dots, M_k\}$ simple vs complex
 $M: y = f_\theta(x) = f_\ell(\cdot|\theta_\ell) \circ \dots \circ f_1(x|\theta_1)$, $\theta = (\theta_1, \dots, \theta_\ell)$
penalty on complexity of M : ℓ_1 , ℓ_2 , dropout (ℓ_2)
- Many other hyper-parameters.

Check statistical stability and reproducibility



Recall: simple model vs. complex model

dataset size n and model complexity (number of parameters p)

- **large- p -small- n** (a.k.a. HDLSS) problems
- impose regularization and sparseness
- Does $\hat{y} = 0.99$ mean that we can be very confident that the associated image input x shows pneumonia? The answer is "No", unless we can provide some statistical justification.³

³bootstrap CI; influence function (functional derivative wrt data dist) for assessing model complexity; Neyman-Pearson test (H_0 : normal vs. H_1 : pneumonia), control type-I, use validation data to assess the testing power

Outline

- 1 Data Problem
- 2 Statistical Models
- 3 Fitting Models (need loss criterion or divergence)
- 4 Model Selection
- 5 Develop One's Own Methodologies**

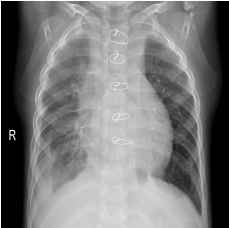
Design and develop one's own methods (day 5)

There are many renowned classical statistical tools for all kinds of data problems. You can incorporate these concepts into your deep model development and design your own methodologies. Here we focus on the following two tools (general purpose).

- **Custom layer for supervised tensor dimension reduction**
- **Robust loss function: γ -divergence**

You might develop other tools for general purpose or specifically for this chest x-ray data problem.

Recall: multilinear PCA

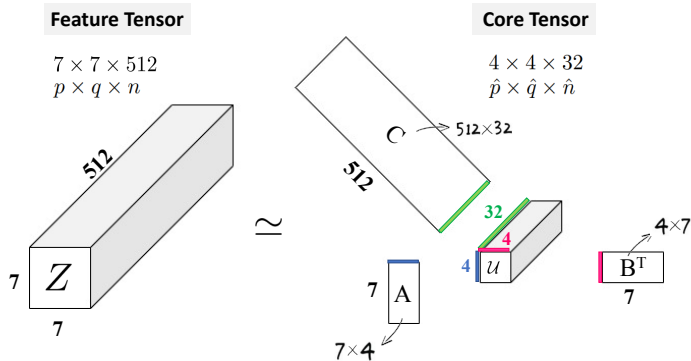
A^T  B logistic regression on $\text{vec}([\cdot]_{5 \times 5})$

$p_1 \times p_2$ reduced to $q_1 \times q_2$
(224×224 to 5×5)

unsupervised dimension reduction

(did not use label information)

TensorProjection layer (for supervised dimension reduction, use label information)



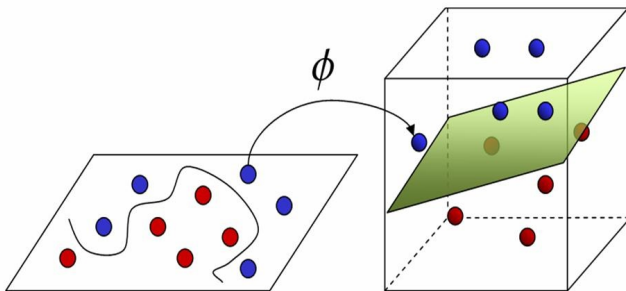
♠ Training data features, which were extracted by vgg16, each has size $7 \times 7 \times 512$

Reduced by TP layer to: $4 \times 4 \times 32$ (code provided: $4 \times 4 \times 16$)

γ -divergence against mislabels and feature outliers

Two types of outlying pattern

- feature outliers
- mislabels



a few words about the custom TP layer

VGG16 + TP layer + regularized logistic regression (model from day 4 code)

```
In [9]: model.summary();
```

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
vgg16 (Model)	(None, 7, 7, 512)	14714688
tensor_projection_layer (Ten	(None, 4, 4, 16)	8248
flatten (Flatten)	(None, 256)	0
dropout (Dropout)	(None, 256)	0
dense (Dense)	(None, 1)	257

```
Total params: 14,723,193
```

```
Trainable params: 8,505
```

```
Non-trainable params: 14,714,688
```

CNN (model from day 2b code)

5 Building the Model

```
In [25]: model = Sequential()
```

```
# 1st Block
```

```
model.add(Conv2D(filters=32, kernel_size=(3,3), strides=(3,3), input_shape=(224,224,3)))  
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
# 2nd Block
```

```
model.add(Conv2D(filters=32, kernel_size=(2,2),strides=(2,2), activation='relu'))  
model.add(BatchNormalization())  
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
# Flatten and Fully Connected Layer
```

```
model.add(Flatten())  
model.add(Dense(100))  
model.add(Activation('relu'))
```

```
# Sigmoid Classifier
```

```
model.add(Dense(1))  
model.add(Activation('sigmoid'))
```

```
# compile model
```

```
model.compile(optimizer='adam',  
              loss='binary_crossentropy',  
              metrics=['accuracy'])
```

Backpropagation in feedforward NN

- sequentially add in layers
- chaining derivatives in backpropagation
- for a custom layer: you will need the gradients of
 - layer output wrt layer input
 - layer output wrt layer parameters
- either (1) automatic differentiation if the required differentials are in the package's tangent library, or (2) you provide these differentials
- In TP layer, parameters are orthogonal matrices (i.e. with constraints $A^T A = I$ and $B^T B = I$)⁴

⁴involve gradients on matrix Stiefel manifold