Weed Classification: Hierarchical Approach

Modeling

In this section, we formulate the hierarchical learning framework for multiclass classification. The intuition is that weed types would be conditional on the weed races based on the Bayesian perspective of probability.

Problem definition

Given an image of weed (or a feature vector extracted from the weed image using any feature extraction method) $\boldsymbol{X_i}$ accompanying with its true label $\boldsymbol{Y_i}$ and true family $\boldsymbol{F_i}, i = 1, \dots, N$ with N is the number of training samples. These labels are one-hot encoding vectors, meaning that $\boldsymbol{Y_i} \in \{0,1\}^M$ and $\boldsymbol{F_i} \in \{0,1\}^K$, M and K is the number of classes and races, respectively. Overall, there are three learning approaches [cite A surevey of hierarchial classification].

$P_W(\widehat{\boldsymbol{Y}}|\boldsymbol{X})$: class predictive model (Flat classifier).

$P_W(\widehat{\boldsymbol{Y}},\widehat{\boldsymbol{F}}|\boldsymbol{X})$: joint class-race predictive model (Big Bang – global classifier).

$P_W(\widehat{\boldsymbol{Y}}|\widehat{\boldsymbol{F}},\boldsymbol{X})$**:** class-conditional-race predictive model (Hierarchically Structured local classifier).

Each model possesses its own learned weights, $\boldsymbol{W}$**.** $\widehat{\boldsymbol{Y}}$ and $\widehat{\boldsymbol{F}}$ are predicted class and race given a feature vector $\boldsymbol{X}$**.** The Directed Graphical model is shown in Figure 1. To derive Maximum Likelihood Estimation (MLE), we consider $\boldsymbol{W}$ as parameters of the predictive models though they can be seen as random variables [cite] as shown in the graphical model.

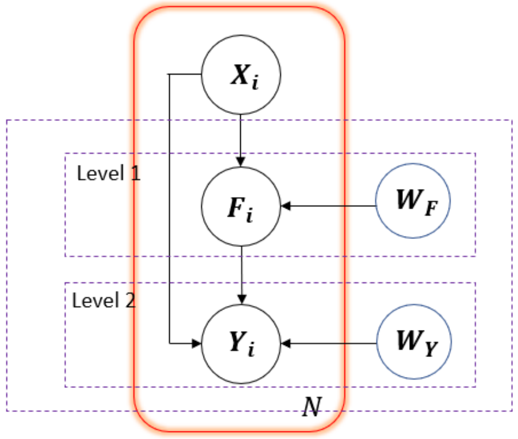

Figure 1. Graphical model of the learning framework. The joint class-race predictive model is presented by the most outside square which contains both race $\boldsymbol{F}_i$ and class $\boldsymbol{Y}_i$ at the same level. While the class predictive model explores the predicted class only (level 2). On the other hand, the hierarchically structured classifier learns in level-order, the class $\boldsymbol{Y}_i$ is distributed conditionally on the race $\boldsymbol{F}_i$.

Objective function

In this work, we estimate the predictive models by using Maximum Likelihood Estimation, a well-defined method to learn the weights of the model. In the multi-classification problem, the predicted labels are usually assumed to be distributed as Multinomial distribution. In the class predictive model, likewise, we assume

$$\widehat{\boldsymbol{Y}}_{\boldsymbol{i}}|\boldsymbol{X}_{\boldsymbol{i}} \sim \boldsymbol{M}(\mu_1, \dots, \mu_M), \boldsymbol{P}\left(\widehat{\boldsymbol{Y}}_{\boldsymbol{i}}|\boldsymbol{X}_{\boldsymbol{i}}\right) = \prod_{k=1}^{M} \mu_k^{\hat{y}_k},$$

Where $0 \leq \mu_k = P\left(\hat{Y}_k = 1|\boldsymbol{X}_{\boldsymbol{i}}\right) = f_W(\boldsymbol{X}_{\boldsymbol{i}}) \leq 1$ is the probability of $\boldsymbol{X}_{\boldsymbol{i}}$ belonging to class k, f_W is an instance in the class of a predefined Neural Network (NN) architecture. The log-likelihood function of all training dataset is

$$L = \log \prod_{i}^{N} \prod_{k}^{M} \mu_{ik}^{\hat{y}_{ik}} = \sum_{i}^{N} \sum_{k}^{M} \hat{y}_{ik} \log(\mu_{ik})$$

Maximizing the log-likelihood function is analogous to minimize the cross-entropy loss function If we make use of the training samples $(\boldsymbol{X}_{\boldsymbol{i}}, \boldsymbol{Y}_{\boldsymbol{i}})$.

$$CrossEntroy = -\sum_{i}^{N} \sum_{k}^{M} y_{ik} \log(\mu_{ik})$$

In the same way, the joint predictive model and class-conditional-race predictive model can be estimated or inferences by assuming the Multinomial distribution. However, there require further processes to derive the above loss function with respect to the global classifier and the hierarchical classifier. The global classifier considers a pair $(\boldsymbol{Y}, \boldsymbol{F})$ simultaneously thus the one-hot coding vector has $M \times K$ components in total, $\boldsymbol{Z} = \boldsymbol{Y} \times \boldsymbol{F}^{\boldsymbol{T}} \in \{0,1\}^{M \times K}$. Meanwhile, the hierarchical classifier provides the information of an input at a higher level before making decision at a lower level. Therefore, it is common that hierarchical learning involves more than one objective term.

One straightforward way to build the hierarchical learning framework is to train the classifier separately from coarse to fine in the level order. In this paper, for example, we trained one model to determine which race a given input belongs to and subsequently trained four sub-models to obtain the specific class. In this way, each model acts as a node in a Decision tree where the parent model identifies the learning route for the child models, Figure 2.

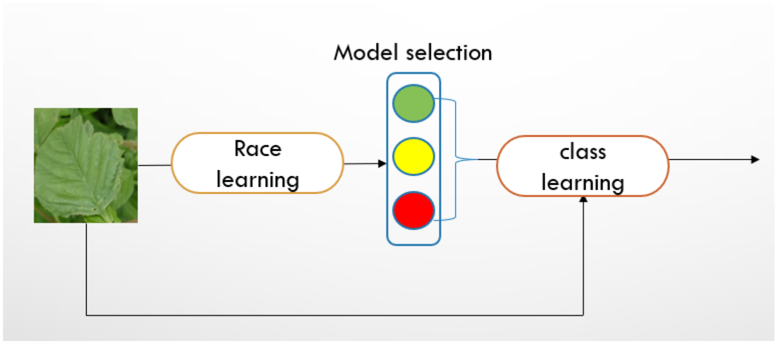

Figure 2. Manual model selection scheme. The given input will be classified into each race before determining the corresponding class.

A necessary condition to efficiently adopt model selection is that the complexity of each model in the hierarchy is lower than that of the single class predictive model. As a result, the computational complexity of the multi-model hierarchy approach increases regarding the depth of the hierarchical level and the number of sub-models at each level. However, an advantage of hierarchical learning is that internal models could be simplified enough to be compatible with a simple task. In this study, we utilize this strategy by selecting simpler models as base classifiers.

Taking advantage of hierarchical features extraction in Deep Convolutional Neural Networks (CNNs), this work presents an intuitive way to learn hierarchical classifiers. It is well known that CNNs analyze a visual entity hierarchically, structuring from shallow layers to deep layers [cite Visualizing and Understanding Convolutional Networks]. Particularly, shallow layers extract global features while deep layers are responsible for learning local features, these features are combined to gain additional details used to recognize an object. The same process is applied to the taxonomic classification of weeds [cite], where the overall shape and color are used to define their families whilst detailed textures are used to specify the classes. The key idea is to extract features at an intermediate layer to calculate the probability of family, which is the evidence to compute the class-conditional on family probability, Figure 3.

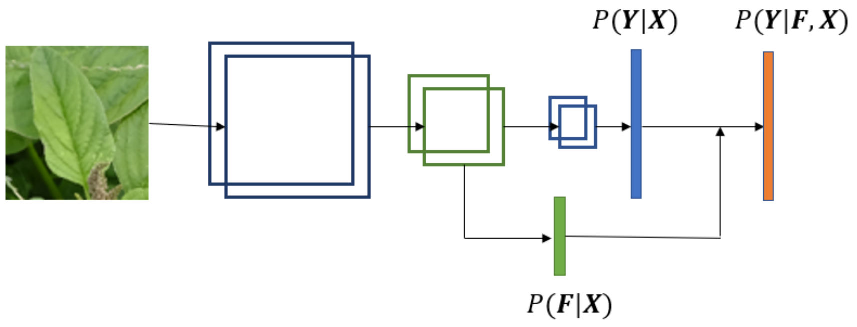
$P(\boldsymbol{Y}|\boldsymbol{X})$
$P(\boldsymbol{Y}|\boldsymbol{F},\boldsymbol{X})$
$P(\boldsymbol{F}|\boldsymbol{X})$

Figure 3. class-conditional on family model.

Formally, we maximize the log-likelihood of class-conditional on family probability, where the probability of $\boldsymbol{X_i}$ belonging to class k given the true family F_j is calculated by the Bayesian formula.

$$\mu_{ik} = P\left(\hat{Y}_k = 1 \middle| \hat{F}_j = 1, X_i\right) = \frac{P\left(\hat{Y}_k = 1, \hat{F}_j = 1 \middle| \boldsymbol{X_i}\right) P\left(\hat{Y}_k = 1 \middle| \boldsymbol{X_i}\right)}{P\left(\hat{F}_j = 1 \middle| \boldsymbol{X_i}\right)},$$

Where j is the true family of input $\boldsymbol{X_i}$. One benefit of this approach is the training and test procedure only requires one feed-forward pass or so-called “end-to-end” model.

A brief overview of ResNet and Mobile Net.

Simply increasing network depth by stacking layers does not imply improving the network's capability because of vanishing gradient problem. This issue makes the gradient such infinitely small that no updates occurring from backpropagation; thus, the performance gets saturated or even worse. To solve the vanishing gradient problem, ResNet [cite ResNet] introduces a special operator called "identity shortcut connection" within a residual block. Instead of learning the desired mapping directly, residual blocks fit a residual mapping, which does not produce a higher training error. As stated by the author, Moreover, simply stacking identity mappings to make the deeper architecture dose not hurt the networks. The great idea of skip connection has been studied extensively in many fields such as segmentation [U-Net,], object detection [YoLo, SSD], and super-resolution []. In this study, we mainly use ResNet18 and ResNet34 as baseline model; we also adopt some pruned architectures to build internal models inside the hierarchy.

A strong point of MobileNet [cite MobileNet version 1] is parameters reduction. As a result, MobileNet is particularly compatible with embedded systems and mobile devices because of its smaller complexity compared to various architectures achieving the same performance. The numbers of multiplication and addition operators in MobileNet could be from 8 to 9 times less than conventional architecture by means of Depth-wise Separable Convolution. Depth-wise Separable Convolution performs spatial convolutional on each channel (depth-wise convolution) followed by 1x1 convolution to change the depth dimension (point-wise convolution). Additionally, MobileNet version 2 [cite MobileNet version 2] addresses the linear bottleneck problem by introducing the Inverted Residuals layer, which widens the channels before embedding. In this work, we manipulate smaller versions of standard MobileNet_v2 to build the base classifiers.

Experiments and results

Recently, instead of randomly initializing weights in Neural Network, the Deep Learning community suggested using the weights which were trained on the large dataset according to their problem [cite A Survey on Transfer Learning]. This study also used the pre-trained weights on ImageNet dataset [cite ImageNet database]. Particularly, ImageNet dataset contains around 999k images and 339k images relating to plants and flowers, respectively. Accordingly, this work adapted transfer learning techniques to reduce the training time and lack of dataset problem. Furthermore, we only used the pre-trained weights as initialized ones, and there is no freezing layer in the Network.

To guarantee the consistency among training processes, we apply the same settings to all models. Specifically, the maximum of epochs is set to 100, the batch size per feed-forward pass is 32, and the final set of weights is selected based on the highest accuracy on the validation set. We train models with SGD algorithm, the learning rate commences at 0.01 and unchanged during the training process.

Table 1. Experiments with ResNet

Residual block	Baseline $(P(Y\|X))$	Family $(P(F\|X))$	Family 1	Family 3	Family 4	Family 5	End-to-End $(P(Y\|F,X))$
			Architecture				
$\begin{bmatrix} 3\times3, & 64 \\ 3\times3, & 64 \end{bmatrix}$	$\times 3$	$\times 2$	$\times 3$	$\times 3$	$\times 2$	$\times 2$	$\times 2$
$\begin{bmatrix} 3\times3, & 128 \\ 3\times3, & 128 \end{bmatrix}$	$\times 4$	$\times 2$	$\times 4$	$\times 4$	$\times 2$	$\times 2$	$\times 2$

$\begin{bmatrix}3\times 3, & 256\\ 3\times 3, & 256\end{bmatrix}$	$\times 6$	$\times 2$	$\times 6$	$\times 0$	$\times 2$	$\times 0$	$\times 2$
$\begin{bmatrix}3\times 3, & 512\\ 3\times 3, & 512\end{bmatrix}$	$\times 3$	$\times 0$	$\times 0$	$\times 0$	$\times 0$	$\times 0$	$\times 2$
Average pooling	$\times 1$						
Output size	[512, 1, 1]	[256, 4, 4]	[256, 4, 4]	[128, 4, 4]	[256, 4, 4]	[128, 4, 4]	[512, 1, 1] [512, 4, 4]
Pretrained weights	ResNet34	ResNet18	ResNet34	ResNet34	ResNet18	ResNet18	ResNet18
Total parameters							
	21,298,010	2,807,366	2,831,948	1,354,051	2,807,366	689,219	11,228,250
Accuracy on test set (%)							
	96.2718	99.5364	95.7839	98.7778	96.1307	99.0921	96.0137

In terms of implementation, a single Residual block basically consists of 2 contiguous convolutional layers with low dimensional embedding (Conv2D → BatchNorm → ReLu → Conv2D → BatchNorm) [cite ResNet]. In each model, the final fully connected layer is replaced by a fully connected layer whose output based on the desired number of labels. Table 1 summarizes the architectures of ResNet used in our experiments and their test accuracies. The baseline class predictive model achieves the accuracy of 96.2718% using ResNet34 with the highest number of parameters, approximately 21M. The ResNet-based hierarchical learning gains an accuracy of **95.6122%,** and the total number of parameters is around **10.5M**, two times smaller than the baseline model.

Table 2. Experiments with MobileNet version 2

Inverted Residual block	Baseline $(P(Y \mid X))$	Family $(P(F \mid X))$	Family 1	Family 3	Family 4	Family 5	End-to-End $(P(Y \mid F, X))$
Architecture							
$\begin{bmatrix}1\times 1, & n\\ 3\times 3, & n\end{bmatrix}$	Full	$\times 15$	$\times 16$	$\times 14$	$\times 16$	$\times 14$	$\times 17$
Output size	[1250, 4, 4]	[160, 4, 4]	[160, 4, 4]	[160, 4, 4]	[160, 4, 4]	[160, 4, 4]	[160, 4, 4] [320, 4, 4]
Total parameters							
	2,756,378	1,084,378	1,368,524	705,475	1,033,158	705,475	1,932,058
Accuracy on test set (%)							
	95.4513	99.5483	97.7796	98.3333	95.9302	99.3515	96.7569

Concerning to implementation, a single Inverted Residual block basically consists of 2 contiguous convolutional layers with high dimensional embedding (Conv2D 1x1 → BatchNorm → ReLu6 → Conv2D → BatchNorm) [cite MbNet v2]. Accompanying with replacing the last fully connected layer, we also drop the last convolutional layer since the desired number of output channels (dimension of embedded vectors) is not appropriate with our small number of classes. Table 2 shows the details of the varied pruned version of MobileNet version 2 used in the experiments. As depicted in the table, the accuracy of the MobileNet baseline model is 95.4513%, lower than the accuracy of the ResNet baseline model. However, the number of parameters is much smaller than the ResNet counterparts. Interestingly, The MobileNet-based hierarchical classifier attains an accuracy of **97.2349 %,** and the total number of consumed parameters in this model is around **4.9M**. four times smaller than the ReseNet baseline model.

The class conditional family predictive model jointly optimizes the objective function with regard to the posterior probability $P(\boldsymbol{Y}|\boldsymbol{F},\boldsymbol{X})$. In another word, there requires only one feed-forward pass to get both class and family probabilities.The Bayesian formula is subsequently applied to increase the confident of prediction. In ResNet-based model, the "family feature" vector is extracted right after the third Residual block, while it is extracted at the 15^{th} Inverted Residual block in the MobileNet-based model. The accuracy of this approach for ResNet-based and MobileNet-based model is **96.0137**% and **96.7569 %**, respectively. Also, This end-to-end model achieves the better accuracy in categorizing the family ($P(\boldsymbol{F}|\boldsymbol{X})$). The ResNet-based model has the accuracy of **99.7443**%, while it is **99.8064**% for MobileNet-based network.

Last but not least, Although the two-level-hierarchical framework takes more than one feed-forward pass, the (shallow) less complexity of basic models doest not explode in the processing time. Considering the average processing time, the MobileNet-based model spends 9ms, while the ResNet-based model demands 4.5 ms to recognize an image with the size of 128x128x3.