Mathematical Foundations of ML - Basic models and statistics

We have a binary classification problem.

We build a predictive model for the problem based on <u>training</u> data.

Example for problem: (x_i, y_i), $x_i \in \mathbb{R}^n$ - vector of patient data, $y_i \in \{0,1\}$ - response variable = {cancer,no cancer}

Option I linear regression. We train for unknown parameters $\theta = (\beta, \beta_0)$, weights $\beta = (\beta_1, \ldots, \beta_n) \in \mathbb{R}^n$ and bias $\beta_0 \in \mathbb{R}$

$$\hat{y}_i = \langle \beta, x_i \rangle + \beta_0 = \sum_{j=1}^{n} \beta_j x_{i,j} + \beta_0 .$$

We solve using the training data

Loss Function

$$\mathrm{Loss}\left(Y \mid X, \tilde{\theta}\right) := \frac{1}{\# I} \sum_{i \in I} \left(\sum_{j=1}^{n} \tilde{\beta}_j x_{i,j} + \tilde{\beta}_0 - y_i \right)^2 ,$$

$$\theta = \arg\min_{\tilde{\theta}} \mathrm{Loss}\left(Y \mid X, \tilde{\theta}\right)$$

Solution is by simple differentiation to find the minima. We get a linear system of dimension $n+1$. For $1 \le k \le n$,

$$\frac{\partial}{\partial \beta_k} \sum_i \left(\sum_{j=1}^{n} \beta_j x_{i,j} + \beta_0 - y_i \right)^2 = 2 \sum_i \left(\sum_{j=1}^{n} \beta_j x_{i,j} + \beta_0 - y_i \right) x_{i,k} = 0 \Leftrightarrow$$
$$\sum_i \left(\sum_{j=1}^{n} \beta_j x_{i,j} x_{i,k} + \beta_0 x_{i,k} - y_i x_{i,k} \right) = 0 \Leftrightarrow$$
$$\sum_{j=1}^{n} \left(\sum_i x_{i,j} x_{i,k} \right) \beta_j + \left(\sum_i x_{i,k} \right) \beta_0 = \sum_i y_i x_{i,k}$$

$$\frac{\partial}{\partial \beta_0} \sum_i \left(\sum_{j=1}^{n} \beta_j x_{i,j} + \beta_0 - y_i \right)^2 = 2 \sum_i \left(\sum_{j=1}^{n} \beta_j x_{i,j} + \beta_0 - y_i \right) = 0 \Leftrightarrow$$
$$\sum_{j=1}^{n} \left(\sum_i x_{i,j} \right) \beta_j + (\# i) \beta_0 = \sum_i y_i$$

Then, for a new incoming data point $x \in \mathbb{R}^n$, we compute (in a regression problem)

$$\hat{y}' = \sum_{j=1}^{n} \beta_j x_j + \beta_0 .$$

For binary classification we perform simple "binning" (two bins in this example)

$$\hat{y} := \begin{cases} 0 & \hat{y}' < 0.5 \\ 1 & \hat{y}' \ge 0.5 \end{cases} .$$

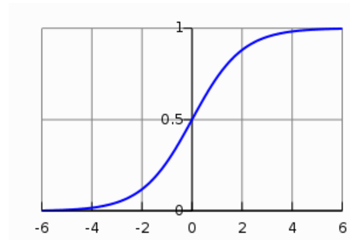
From statistical viewpoint - we have not utilized the fact that the problem is "categorical" binary classification.

From approximation theoretical perspective – We have not utilized the fact that we want to approximate a piecewise constant function with a boundary determined by a hyperplane.

Logistic regression and SoftMax loss function

The logistic function

$$\sigma(t) := \frac{1}{1+e^{-t}} \; .$$



$\sigma(t) \underset{t\to-\infty}{\to} 0$, $\sigma(0) = 0.5$, $\sigma(t) \underset{t\to\infty}{\to} 1$

We then model with $\theta := \{\beta, \beta_0\} \in \mathbb{R}^{n+1}$

$$\hat{y}' = h_\theta(x) := \frac{1}{1+e^{-(\langle \beta, x\rangle + \beta_0)}} \; .$$

Statistical Modeling: $\Pr(y \mid x, \theta) = h_\theta(x)^y \left(1 - h_\theta(x)\right)^{1-y}$

So we want to maximize the likelihood function

$$\begin{aligned} L(\theta \mid x) &= \Pr(Y \mid X, \theta) \\ &= \prod_{i\in I} \Pr(y_i \mid x_i, \theta) \\ &= \prod_{i\in I} h_\theta(x_i)^{y_i} \left(1 - h_\theta(x_i)\right)^{1-y_i} \end{aligned}$$

One typically normalizes with the size of the data set and minimizes the negative log-likelihood

$$\begin{aligned} -\frac{1}{\# I}\log L(\theta \mid x) &= -\frac{1}{\# I}\log \prod_{i\in I} h_\theta(x_i)^{y_i} \left(1 - h_\theta(x_i)\right)^{1-y_i} \\ &= -\frac{1}{\# I}\sum_{i\in I} y_i \log h_\theta(x_i) + (1-y_i)\log\left(1 - h_\theta(x_i)\right) \end{aligned}$$

This could be used on a pixel by pixel basis: the pixel is part of a segmentation or not.

Minimization via gradient descent methods (not a linear system like linear regression).

Also, this could be used for a multi-class problem where an image can have more than one label.

Suppose we have L classes. Each training vector x_i has L labels $y_{i,k}$, $k = 1, \ldots, L$. The modeling is done through $\theta_k = \left(\beta^k, \beta_0^k\right)$

$$\begin{aligned} L(\theta \mid x) &= \Pr(Y \mid X, \theta) \\ &= \prod_{i\in I}\prod_{k=1}^{L} \Pr(y_{i,k} \mid x_i, \theta_k) \\ &= \prod_{i\in I}\prod_{k=1}^{L} h_{\theta_k}(x_i)^{y_{i,k}} \left(1 - h_{\theta_k}(x_i)\right)^{1-y_{i,k}} \end{aligned}$$

$$-\frac{1}{(\#I)L}\log L(\theta|x) = -\frac{1}{(\#I)L}\log\prod_{i\in I}\prod_{k=1}^{L} h_{\theta_k}(x_i)^{y_{i,k}}\left(1-h_{\theta_k}(x_i)\right)^{1-y_{i,k}}$$
$$= -\frac{1}{(\#I)L}\sum_{i\in I}\sum_{k=1}^{L} y_{i,k}\log h_{\theta_k}(x_i) + \left(1-y_{i,k}\right)\log\left(1-h_{\theta_k}(x_i)\right)$$

This is separable (!) if there is no additional joint architecture that creates the feature space X ! That is, one could minimize separately for $1 \le k \le L$,

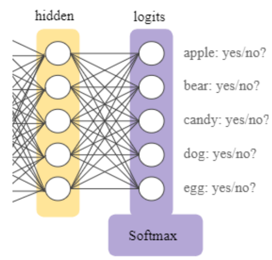
$$-\frac{1}{(\#I)}\sum_{i\in I} y_{i,k}\log h_{\theta_k}(x_i) + \left(1-y_{i,k}\right)\log\left(1-h_{\theta_k}(x_i)\right).$$

Examples of multi-class labels in computer vision: woman/girl/dress, etc.

Soft-Max

If we have a classification problem which is not (!) a multi-class problem and we want to encourage a choice, we apply a soft-max technique. We use $\theta = \left\{W \in M_{L\times n}, b \in \mathbb{R}^L\right\}$. We model, with w_k, the k-th row of W, $1 \le k \le L$,

$$\Pr\left(y = Y_k \mid x, \theta\right) := \frac{e^{\left(\langle W_k, x\rangle + b_k\right)}}{\sum_{j=1}^{L} e^{\left(\langle W_j, x\rangle + b_j\right)}}$$



The associated loss function is the minimization of the negative of the log-likelihood

$$-\frac{1}{\#I}\sum_{i\in I}\log\left(\Pr\left(Y = y_i \mid \theta, x_i\right)\right).$$

<u>Basic Definitions</u>

<u>Training set</u> – We train the model using training data

<u>Validation set</u> - We have the option of using some ‘holdout’ data for hyper-parameter tuning

<u>Testing set</u> – We use ‘ground truth’ testing data to analyze the performance of the model.

Typical example – 70% training, 10% validation, 20% testing

<u>Cross validation</u> – A research technique where we randomly split the full set into (for example) 5 parts, build a model 5 times, each time testing on a different part after training on the remaining 4 parts. One can also randomly split the full set 5 times. We then average the error/accuracy by each model.

<u>Inference</u> –In applications, we apply the model to incoming unlabeled data.

Confusion matrix

	Actual	
Predicted		**Categorical**	**Non-Categorical**
	Categorical	True Positive (TP)	False Positive (FP)
	Non-Categorical	False Negative (FN)	True Negative (TN)

FP = Type I Error, FN = Type II Error

We can obviously create a confusion matrix for arbitrary number of classes

		Actual				
Predicted		Class 1	Class 2	…	…	Class L
	Class 1					
	Class 2					
	Class L					

Optimally, outside the diagonal we hope to get small numbers/%.

Definition: Accuracy. The simplest form of measurement

$$\frac{TP+TN}{P+N} = \frac{TP+TN}{(TP+FN)+(TN+FP)}$$

Accuracy is problematic in "rare events cases". Suppose that we have a positives for 0.1% of the time. Then, a 'stupid' model that predicts Negtive for each sample has accuracy 99.9%.

Definitions

Sensitivity, Recall, True Positive Rate $\frac{TP}{P} = \frac{TP}{TP+FN}$

e.g.: what % of the cancer patients did the model find?

e.g. in document retrieval : relevant and retrieved / relevant

The 'stupid' model that always predicts Negative will have recall = 0.

Specificity, True Negative (False) Rate $\frac{TN}{N} = \frac{TN}{TN+FP}$

Averaged accuracy $\frac{1}{2}\left(\frac{TP}{P} + \frac{TN}{N}\right)$

Precision (Positive Prediction Value) $\frac{TP}{TP+FP}$

e.g in document retrieval : Relevant and retrieved / all retrieved

False Positive Rate (fall out, false alarms) $\frac{FP}{N} = \frac{FP}{FP + TN}$

False Negative Rate (miss rate) $\frac{FN}{P} = \frac{FN}{TP + FN}$

F1 Score $2\frac{precision \times recall}{precision + recall}$