1.1.Từ Binary classification tới multi-class classification

Các phương pháp Support Vector Machine đã đề cập (Hard Margin, Soft Margin, Kernel) đều được xây dựng nhằm giải quyết bài toán [Binary Classification](https://machinelearningcoban.com/2017/02/11/binaryclassifiers/), tức bài toán phân lớp với chỉ hai classes. Việc này cũng tương tự như [Percetron Learning Algorithm](https://machinelearningcoban.com/2017/01/21/perceptron/) hay [Logistic Regression](https://machinelearningcoban.com/2017/01/27/logisticregression/) vậy. Các mô hình làm việc với bài toán có 2 classes còn được gọi là Binary classifiers. Một cách tự nhiên để mở rộng các mô hình này áp dụng cho các bài toán multi-class classification, tức có nhiều classes dữ liệu khác nhau, là [sử dụng nhiều binary classifiers và các kỹ thuật như one-vs-one hoặc one-vs-rest](https://machinelearningcoban.com/2017/02/11/binaryclassifiers/#-binary-classifiers-cho-multi-class-classification-problems). Cách làm này có những hạn chế như đã trình bày trong bài [Softmax Regression](https://machinelearningcoban.com/2017/02/17/softmax/).

Softmax Regression là mở rộng của Logistic Regression cho bài toán multi-class classification, có thể được coi là một layer của Neural Networks. Nhờ đó, Softmax Regression thường đươc sử dụng rất nhiều trong các bộ phân lớp hiện nay. Các bộ phân lớp cho kết quả cao nhất thường là một Neural Network với rất nhiều layers và layer cuối là một softmax regression, đặc biệt là các Convolutional Neural Networks. Các layer trước thường là kết hợp của các Convolutional layers và các nonlinear activation functions và pooling, các bạn tạm thời chưa cần quan tâm đến các layers phía trước này, tôi sẽ giới thiệu khi có dịp. Có thể coi các layer trước layer cuối là một công cụ giúp trích chọn đặc trưng của dữ liệu (Feature extraction), layer cuối là softmax regression, là một bộ phân lớp tuyến tính đơn giản nhưng rất hiệu quả. Bằng cách này, ta có thể coi là nhiều one-vs-rest classifers được huấn luyện cùng nhau, hỗ trợ lẫn nhau, vì vậy, một cách tự nhiên, sẽ có thể tốt hơn là huấn luyện từng classifier riêng lẻ.

Sự hiệu quả của Softmax Regression nói riêng và Convolutional Neural Networks nói chung là cả *bộ trích chọn đặc trưng* (feature extractor) và *bộ phân lớp* (classifier) được *huấn luyện* đồng thời. Điều này nghĩa là hai *bộ phận* này bổ trợ cho nhau trong quá trình huấn luyện. Classifier giúp tìm ra các hệ số hợp lý phù hợp với feature vector tìm được, ngược lại, feature extractor lại điều chỉnh các hệ số của các convolutional layer sao cho feature thu được là tuyến tính, phù hợp với classifier ở layer cuối cùng.

Tôi viết đến đây không phải là để giới thiệu về Softmax Regression, mà là đang nói chung đến các mô hình phân lớp *hiện đại*. Đặc điểm chung của chúng là feature extractor và classifier được huấn luyện một cách đồng thời. Những mô hình như thế này còn được gọi là *end-to-end*. Cùng xem lại mô hình chung cho các bài toán Machine Learning mà tôi đã đề cập trong Bài 11:



Trong Hình 1, phần TRAINING PHASE, chúng ta có thể thấy rằng có hai khối chính là *Feature Extraction* và *Classification/Regression/Clustering…* Các phương pháp *truyền thống* thường xây dựng hai khối này qua các bước riêng rẽ. Phần Feature Extraction với dữ liệu ảnh có thể dùng các feature descriptor như [SIFT](http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html), [SURF](http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html), [HOG](http://www.learnopencv.com/histogram-of-oriented-gradients/); với dữ liệu văn bản thì có thể là [Bag of Words](https://machinelearningcoban.com/2017/04/28/multiclasssmv/2017/02/06/featureengineering/#bag-of-words) hoặc [TF-IDF](http://www.tfidf.com/). Nếu là các bài toán classification, phần còn lại có thể là SVM thông thường hay các bộ phân lớp *truyền thống* khác.

Với sự phát triển của Deep Learning trong những năm gần đây, người ta cho rằng các hệ thống *end-to-end* (từ đầu đến cuối) mang lại kết quả tốt hơn nhờ và việc các hai khối phía trên được huấn luyện cùng nhau, bổ trợ lẫn nhau. Thực tế cho thấy, các phương pháp *state-of-the-art* thường là các mô hình *end-to-end*.

Các phương pháp Support Vector Machine được chứng minh là tốt hơn Logistic Regression vì chúng có quan tâm đến việc tạo *margin* lớn nhất giữa các classes. Câu hỏi đặt ra là:

**Liệu có cách nào giúp kết hợp SVM với Neural Networks để tạo ra một bộ phân lớp tốt với bài toán multi-class classification? Hơn nữa, toàn bộ hệ thống có thể được huấn luyện theo kiểu *end-to-end*?**

Câu trả lời sẽ được tìm thấy trong bài viết này, bằng một phương pháp được gọi là *Multi-class Support Vector Machine*.

Và để cho bài viết hấp dẫn hơn, tôi xin giới thiệu luôn, ở phần cuối, chúng ta sẽ cùng lập trình từ đầu đến cuối để giải quyết bài toán phân lớp với bộ cơ sở dữ liệu nổi tiếng: CIFAR10.

**1.3. Bộ cơ sở dữ liệu CIFAR10**

Bộ cơ sở dữ liệu CIFAR10 gồm 51000 ảnh khác nhau thuộc 10 classes: *plane, car, bird, cat, deer, dog, frog, horse, ship, và truck*. Mỗi bức ảnh có kích thước 32×3232×32 pixel. Một vài ví dụ cho mỗi class được cho trong Hình 2 dưới đây. 50000 ảnh được sử dụng cho training, 1000 ảnh còn lại được dùng cho test. Trong số 50000 ảnh training, 1000 ảnh sẽ được lấy ra ngẫu nghiên để làm [validation set](https://machinelearningcoban.com/2017/03/04/overfitting/#-validation-1).

Graphical user interface

Description automatically generated with medium confidence

Đây là một bộ cơ sở dữ liệu tương đối khó vì ảnh nhỏ và object trong cùng một class cũng biến đổi rất nhiều về màu sắc, hình dáng, kích thước. [Thuật toán tốt nhất hiện nay cho bài toán này](http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html#43494641522d3130) đã đạt được độ chính xác trên 90%, sử dụng một Convolutional Neural Network nhiều lớp kết hợp với Softmax regression ở layer cuối cùng. Trong bài này, chúng ta sẽ sử dụng một mô hình neural network đơn giản không có hidden layer nào để giải quyết, kết quả đạt được là khoảng 40%, nhưng cũng là đã rất ấn tượng. Layer cuối là một layer Multi-class SVM. Tôi sẽ hướng dẫn các bạn lập trình cho mô hình này từ đầu đến cuối mà không sử dụng một thư viện đặc biệt nào ngoài numpy.

Bài toán này cũng như nội dung chính của bài viết được lấy từ Lecture notes: [Linear Classifier II](http://cs231n.github.io/linear-classify/) và [Assignment #1](http://cs231n.github.io/assignments2016/assignment1/) trong khoá học [CS231n: Convolutional Neural Networks for Visual Recognition](http://cs231n.github.io/) kỳ Winter 2016 của Stanford.

Trước khi đi vào mục xây dựng hàm mất mát cho Multi-class SVM, tôi muốn nhắc lại một chút về một chút feature engineering cho ảnh trong CIFAR-10 và **bias trick** nói chung trong Neural Networks.

1.4. Image data preprocessing

Để cho mọi thứ được đơn giản và có được một mô hình hoàn chỉnh, chúng ta sẽ sử dụng phương pháp *feature engineering* đơn giản nhất: lấy trực tiếp tất cả các pixel trong mỗi ảnh và thêm một chút normalization.

* Mỗi ảnh của CIFAR-10 đã có kích thước giống nhau 32×3232×32 pixel, vì vậy việc đầu tiên chúng ta cần làm là *kéo dài* mỗi trong ba channels Red, Green, Blue của bức ảnh ra thành một vector có kích thước là 3×32×32=30723×32×32=3072.
* Vì mỗi pixel có giá trị là một số tự nhiên từ 0 đến 255 nên chúng ta cần một chút [chuẩn hóa dữ liệu](https://machinelearningcoban.com/general/2017/02/06/featureengineering/#feature-scaling-and-normalization). Trong Machine Learning, một cách đơn giản nhất để chuẩn hóa dữ liệu là **center data**, tức làm cho mỗi feature có trung bình cộng bằng 0. Một cách đơn giản để làm việc này là ta tính trung bình cộng của tất cả các ảnh trong tập training để được *ảnh trung bình*, sau đó trừ từ tất cả các ảnh đi *ảnh trung bình* này. Tương tự, ta cũng dùng *ảnh trung bình* này để chuẩn hoá dữ liệu trong *validation set* và *test set*.

A picture containing diagram

Description automatically generated

Bây giờ thì ta chỉ còn 1 biến dữ liệu là W thay vì hai biến dữ liệu như trước. Từ giờ trở đi, khi viết W và x, chúng ta ngầm hiểu là biến mới và dữ liệu mới như ở phần bên phải của Hình 3.

**2. Xây dựng hàm mất mát cho Multi-class Support Vector Machine**

Chúng ta cùng quay lại một chút với ý tưởng của Softmax Regression với hàm mất mát Cross-entropy. Sau đó, chúng ta sẽ làm quen với Multi-class SVM với hàm mất mát hinge loss mở rộng.

**2.1. Nhắc lại Softmax Regression.**

Chúng ta cùng xem lại [Softmax layer đã được trình bày trong Bài 13](https://machinelearningcoban.com/2017/02/17/softmax/#-softmax-function).

Background pattern

Description automatically generated with low confidence

Trong Hình 4 ở trên, dữ liệu trong lớp màu xanh lục được coi như *feature vector* của dữ liệu. Với dữ liệu CIFAR-10, nếu ta coi mỗi feature là giá trị của từng pixel trong ảnh, tổng số chiều của *feature vector* cho mỗi bức ảnh là 32×32×3+1=307332×32×3+1=3073, với 3 là số channels trong bức ảnh (Red, Green, Blue).

Qua ma trận hệ số W, dữ liệu ban đầu trở thành z=wT x

Lúc này, ứng với mỗi một trong C classes, chúng ta nhận được một giá trị tương ứng zi ứng với class thứ i. Giá trị zi này còn được gọi là *score* của dữ liệu x ứng với class thứ i.

Ý tưởng chính trong Softmax Regression là đi tìm ma trận hệ số W, mỗi cột của ma trận này ứng với một class, sao cho *score vector* z đạt giá trị lớn nhất tại phần tử tương ứng với class chính xác của nó. Sau khi mô hình đã được *trained*, *nhãn* của một điểm dữ liệu mới được tính là vị trí của thành phần score có giá trị lớn nhất trong *score vector*. Xem ví dụ trong Hình 5 dưới đây:

A picture containing table

Description automatically generated

Hình 5: Ví dụ về cách tính score vector. Khi test, nhãn của dữ liệu được xác định dựa trên class có score cao nhất.

Để huấn luyện trên tập các cặp (*dữ liệu*, *nhãn*), Softmax Regression sử dụng hàm softmax để đưa *score vector* về dạng phân phối xác suất có các phần tử là dương và có tổng bằng 1. Sau đó dùng hàm cross entropy để *ép* vector xác suất này gần với vector xác suất *thật sự* của dữ liệu - tức one-hot vector mà chỉ có đúng 1 phần tử bằng 1 tại class tương ứng, các phần tử còn lại bằng 0.

2.3. Hinge losss tổng quát cho Multi-class SVM

Với Multi-class SVM, trong khi tesst, class của một input cũng được xác định bởi thành phần có giá trị lớn nhất trong score vector. Điều này giống với Softmax Regression.

Softmax Regression sử dụng cross-entropy để *ép* hai vector xác suất bằng nhau, tức ép phần tử tương ứng với *correct class* trong vector xác suất gần với 1, đồng thời, các phần tử còn lại trong vector đó gần với 0. Nói cách khác, cách làm này khiến cho phần tử tương ứng với *correct class* càng lớn hơn các phần tử còn lại càng tốt. Trong khi đó, Multi-class SVM sử dụng một chiến thuật khác cho mục đích tương tự dựa trên *score vector*. Điểm khác biệt là Multi-class SVM xây dựng hàm mất mát dựa trên định nghĩa của *biên an toàn*, giống như trong Hard/Soft Margin vậy. Multi-class SVM *muốn* thành phần ứng với *correct class* của *score vector* lớn hơn các phần tử khác, không những thế, nó còn lớn hơn một đại lượng Δ>0 gọi là *biên an toàn*. Hãy xem Hình 6 dưới đây:

Diagram, schematic

Description automatically generated

Hình 6: Mô tả hinge loss cho Multi-class Support Vector Machine. Multi-class SVM muốn score của correct class , được minh hoạ bởi điểm màu lam, cao hơn các scores khác, minh hoạ bởi các điểm màu lục, một khoảng cách an toàn Δ là đoạn màu đỏ. Những scores khác nằm trong vùng an toàn (phía trái của điểm x màu đỏ) sẽ không gây ra mất mát gì, những scores nằm trong hoặc bên phải vùng màu đỏ đã vi phạm quy tắc và cần được xử phạt.

**2.6. Soft Margin SVM là một trường hợp đặc biệt của Multi-class SVM**

Phát biểu này có vẻ hiển nhiên vì bài toán phân lớp với hai classes là một trường hợp đặc biệt của bài toán phân lớp với nhiều classes! Nhưng điều tôi muốn nói đến là cách xây dựng hàm mất mát. Điều này có thể được nhận ra bằng cách xét từng điểm dữ liệu.