Linear classification

Score function: maps the raw data to class scores
Loss function: quantifies the agreement between the predicted and he ground truth labels

Parameterized mapping from images to label scores: $f : R^D \mapsto R^K$, where D is the dimension of one input and K is the number of classes. In other words, the function maps the raw image pixels to class scores.

Linear classifier: $f(x_i,\ W,\ B)\ =\ Wx_i + b$

- x_i : flattened image
- b: bias
- W: weights
- Change the w rows will rotate the line that separates (classifies) the images.
- Change b will translate the lines
- Interpretation of linear classifiers as a template matching: each row of W corresponds to a template for one of the classes and the score is obtained comparing each template with the image using an inner product.
- Bias trick: add a additional dimension to vector x_i with constant 1. Also, merge b as the last column of W: $f(x_i,\ W)\ =\ Wx_i$
- Center your data: kind of normalization of input features by subtracting the mean from every feature. Another way of normalization is scaling each input values to range from [-1, 1].

Loss function (cost function or the objective): the loss will be high if we're doing a poor job of classifying the training data, and it will be low if we're doing well.

Multiclass Support Vector Machine Loss: The SVM loss is set up so that the SVM "wants" the correct class for each image to a have a score higher than the incorrect classes by some fixed margin Δ. If this is not the case, we will accumulate loss.

Data loss: $L_i = \sum_{j \neq y_i} max(0,\ s_j - s_{y_i} + \Delta)$

For linear functions: $L_i = \sum_{j \neq y_i} max(0, w_j^T x_i - w_{y_i}^T x_i + \Delta)$

Hinge loss: the threshold at zero max(0,−)

Regularization penalty (R(W)): $R(W) = \sum_k \sum_l W_{k,l}^2$

Full multiclass SVM loss: $L = \frac{1}{N} \sum_i L_i + \lambda R(W)$

- N is the number of training examples.
- λ is the weight that is given to the regularization penalty (determined by cross-validation).
- Penalizing large weights tends to improve generalization, because it means that no input dimension can have a very large influence on the scores all by itself.

- the only real tradeoff is how large we allow the weights to grow (through the regularization strength λ).

Softmax classifier

Cross-entropy loss: $L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_i}}\right) = -f_{y_i} + \log\sum_j e^{f_i}$

Softmax function: $f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}}$

Cross-entropy between a true distribution p and an estimated distribution q:

$H(p,q) = -\sum_x p(x)\log q(x)$

Probabilistic interpretation: $P(y_i|x_i;W) = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$ can be interpreted as the (normalized) probability assigned to the correct label yi given the image xi and parameterized by W.

SVM vs. Softmax: can be interpreted as the (normalized) probability assigned to the correct label yi given the image xi and parameterized by W.