Course Notes: Deep Learning for Visual Computing

Peter Wonka

August 30, 2021

Contents

1 Linear Classification and Loss Functions

1.1 Literature

- Content is a shorter version of EECS 498.007 / 598.005: Deep Learning for Computer Vision Fall 2019, Lecture 3, Linear Classifiers
  - youtube

1.2 Classifier Components

- **Score function** predicts a class label for a given image
- **Loss function** quantifies the (dis)agreement between the predicted scores and the ground truth
- **Optimization** minimizes the loss function with respect to the parameters of the neural network

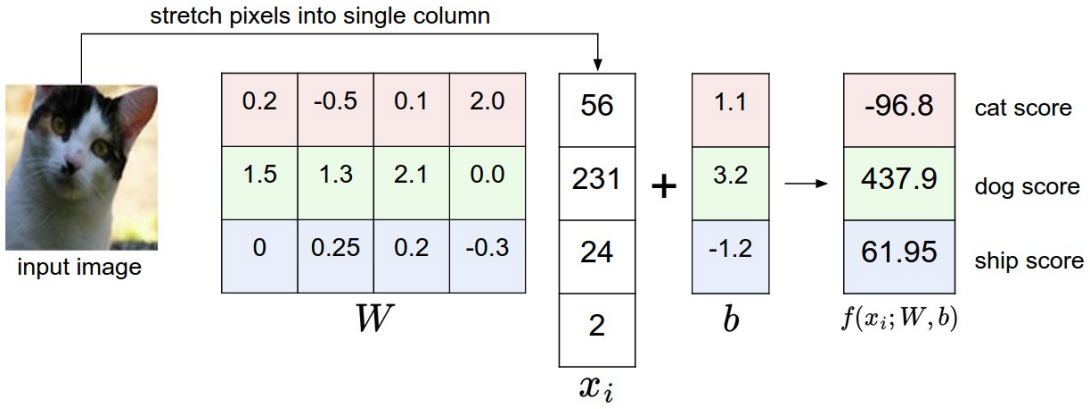
1.3 One hot encoding

- Class labels are originally given as integers, e.g. $\{1, 2, 3, \ldots, K\}$
- One hot encoding encodes a class label i as vectors $(0, \ldots, 1, \ldots, 0)^T$ with the 1 at the ith position
- Better for a probabilistic interpretation and optimization

1.4 Linear Score Function

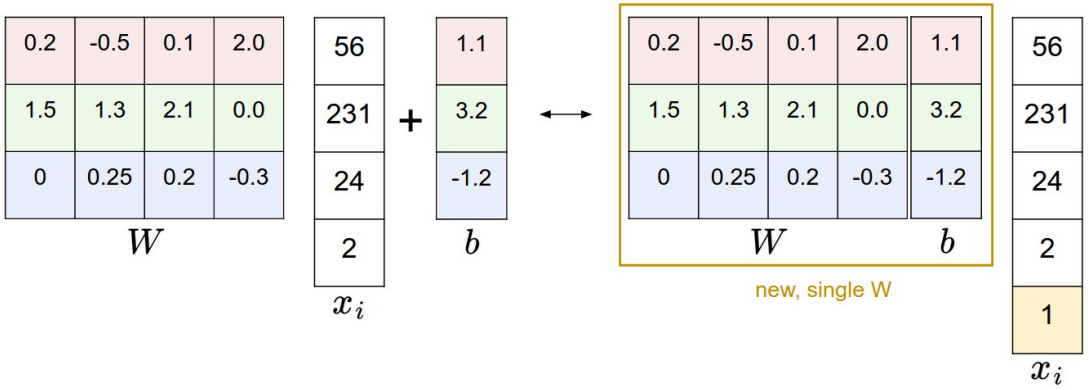
- An image is given as a vector as vector $\mathbf{x}_i$ with D dimensions.
  - e.g. CIFAR-10 image is described by $D = 32 \times 32 \times 3 = 3072$ numbers
  - Left: Cat with 1024×1024 pixels. Right: Cat with 32×32 pixels

- We have a set of images $\mathbf{x}_i$ with corresponding labels y_i from a set of K labels.
  - e.g. CIFAR-10 has $K = 10$
- We use a linear function $f(\mathbf{x}_i; \boldsymbol{W}, \mathbf{b}) : \mathbb{R}^D \rightarrow \mathbb{R}^K$ as score function
- $f(\mathbf{x}_i; \boldsymbol{W}, \mathbf{b}) = \boldsymbol{W}\mathbf{x}_i + \mathbf{b}$
  - images $\mathbf{x}_i \in \mathbb{R}^D$
  - weight matrix $\boldsymbol{W} \in \mathbb{R}^{K \times D}$
  - $\mathbf{b} \in \mathbb{R}^K$
  - output is vector $\in \mathbb{R}^K$
  - each element of the output is a class score for each of the 10 classes (higher means more of the class)
  - basically a neural network consisting of a single linear layer


stretch pixels into single column
input image
0.2
-0.5
0.1
2.0
1.5
1.3
2.1
0.0
0
0.25
0.2
-0.3
W
56
231
24
2
x_i
+
1.1
3.2
-1.2
b
-96.8
437.9
61.95
cat score
dog score
ship score
f(x_i; W, b)

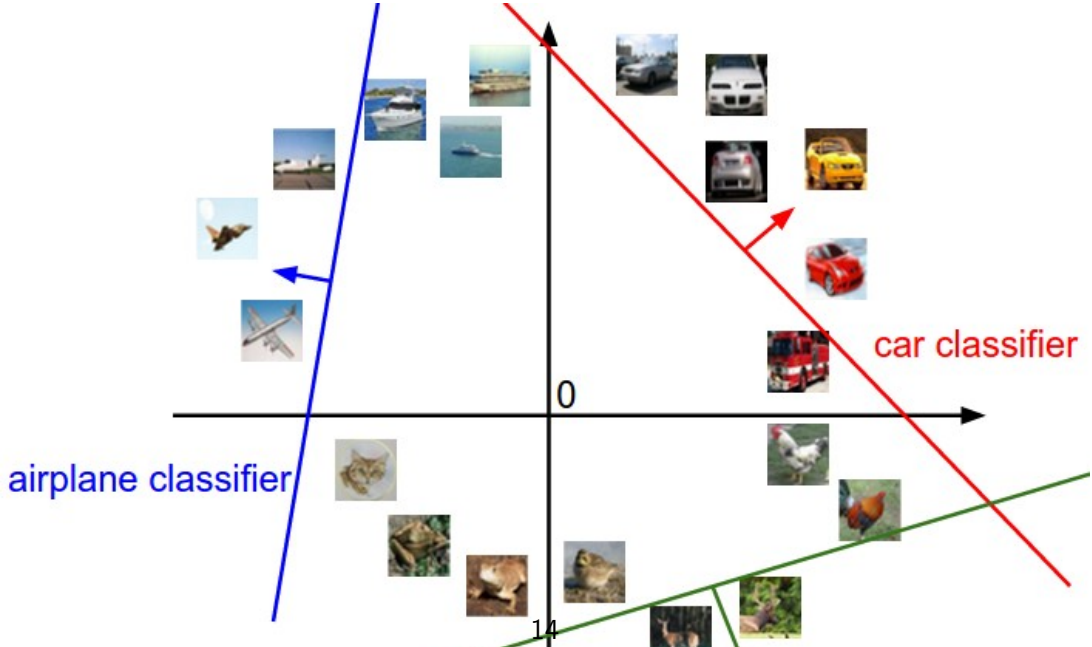
1.5 Simplify Notation

- Simplify Notation by adding an element 1 to $\mathbf{x}_i$ and adding a column to $\boldsymbol{W}$: $f(\mathbf{x}_i; \boldsymbol{W}) = \boldsymbol{W}\mathbf{x}_i$
  - images $\mathbf{x}_i \in \mathbb{R}^{D+1}$
  - weight matrix $\boldsymbol{W} \in \mathbb{R}^{K \times D+1}$
  - output is vector $\in \mathbb{R}^K$
  - previous vector $\mathbf{b}$ is the last column of $\boldsymbol{W}$
- Example: CIFAR-10 would be $f(\mathbf{x}_i; \boldsymbol{W}) : \mathbb{R}^{3072+1} \rightarrow \mathbb{R}^{10}$ as score function
  - $\boldsymbol{W} \in \mathbb{R}^{10 \times (3072+1)}$
- Note: Shortens notation on the slides; We can refer to all network weights as $\mathbf{W}$


0.2 -0.5 0.1 2.0
1.5 1.3 2.1 0.0
0 0.25 0.2 -0.3
W
56
231
24
2
x_i
+
1.1
3.2
-1.2
b
0.2 -0.5 0.1 2.0 1.1
1.5 1.3 2.1 0.0 3.2
0 0.25 0.2 -0.3 -1.2
W b
new, single W
56
231
24
2
1
x_i

1.6 Interpretation as K Classifiers in D dimensional space

- We can images as points in D dimensional space and each row of $\boldsymbol{W}$ as separate linear classifier in high dimensional space
- The i^{th} row of $\boldsymbol{W}$ controls the rotation and offset of the i^{th} hyperplane
- The score on the hyperplane is 0. The arrows in the figure show in what direction the score increases.


car classifier
0
airplane classifier

1.7 Interpretation as K templates

- We can think of the rows of $\boldsymbol{W}$ as template images.
- Templates are also called prototypes.
- Each image is compared to each of the K template using an inner product (dot product).
- Below is a visualization of templates learned for CIFAR-10

- Note the two headed horse template. Note the blue background for planes and ships.

1.8 Loss Function Overview

- We measure our unhappiness with the predicted scores using a **loss function**
  - also called **cost function** or **objective**
- e.g. if an image is a cat we want the predicted cat score to be high and the other scores to be low
- Loss over the data set is the average loss over each sample

$$L = \frac{1}{N} \sum_i L_i(f(\mathbf{x}_i; \boldsymbol{W}), y_i) \tag{1.1}$$

  - L_i is the loss of sample i
  - N is number of samples

1.9 Two Popular Losses

- **Hinge loss** for an SVM Classifier
  - also called **Max-margin loss**
  - also called **Multiclass Support Vector Machine loss**
- **Cross-entropy loss** for a Softmax Classifier
  - also called **Softmax loss**
- Note there are multiple versions of these losses

1.10 Multiclass Support Vector Machine loss

- Terms: **Multiclass Support Vector Machine loss** or **SVM loss**
- SVM loss wants the correct class for each image to a have a score higher than the incorrect classes by some fixed margin Δ
- Scale is typically arbitrary, therefore $\Delta = 1$

- Notation:
  - We call the output of the score function $\mathbf{s} = f(\mathbf{x}_i; \boldsymbol{W})$
  - The j-th element of $\mathbf{s}$ defines the score for the j-th class: $\mathbf{s}_j = f(\mathbf{x}_i; \boldsymbol{W})_j$
    - using $\mathbf{s}_{ij}$, $\mathbf{s}_{\mathbf{i}j}$, or $\boldsymbol{S}_{ij}$ seems more annoying
- SVM loss for image i is:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + \Delta) \tag{1.2}$$

  - Sum over all labels except the ground truth label y_i
  - s_{y_i} is the predicted score of ground truth label y_i
  - Another term for losses of the form $\max(0, ...)$ is **hinge loss**.
- Example Calculation:



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	12.9

- Assume $\Delta = 1$
- Cat column: $\max(0, 5.1-3.2+1)+\max(0, -1.7-3.2+1) = \max(0, 2.9)+\max(0, -3.9) = 2.9$
- Car column: $\max(0, 1.3 - 4.9 + 1) + \max(0, 2.0 - 4.9 + 1) = 0$
- Frog column: $\max(0, 2.2 - (-3.1) + 1) + \max(0, 2.5 - (-3.1) + 1) = 6.3 + 6.6 = 12.9$
- $L = \frac{1}{3}(L_1 + L_2 + L_3) = 15.8/3 = 5.27$

- Questions about the loss function:
  - What is the min/max possible loss?
  - At initialization, all entries in $\boldsymbol{W}$ are small, $\mathbf{s} \approx 0$, what is the loss?
  - Assume we found $\boldsymbol{W}$ such that $L = 0$. Is this $\boldsymbol{W}$ unique?
    - No, $2\boldsymbol{W}$ also has $L = 0$
    - → we want to introduce regularization

1.11 Regularization

- New Loss: Data Loss + Regularization:

$$L = \frac{1}{N}\sum_i L_i(f(\mathbf{x}_i; \boldsymbol{W}), y_i) + \lambda R(\boldsymbol{W}) \tag{1.3}$$

  - λ strength of the regularization
  - $R(\boldsymbol{W})$ regularization
- Simple Regularization Examples:
  - $L2$ regularization: $R(\boldsymbol{W}) = \sum_k \sum_l W_{kl}^2$
  - $L1$ regularization: $R(\boldsymbol{W}) = \sum_k \sum_l |W_{kl}|$
  - Elastic Net $(L1 + L2)$: $R(\boldsymbol{W}) = \sum_k \sum_l \left(W_{kl}^2 + |W_{kl}|\right)$
- Why regularize?
  - Express preferences over weights

- Make the model simple so it works better on test data
- Improve optimization by adding curvature

- $L2$ Regularization Example:
  -
$$\mathbf{x} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \quad \mathbf{w}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{w_2} = \begin{pmatrix} 0.25 \\ 0.25 \\ 0.25 \\ 0.25 \end{pmatrix}, \quad \mathbf{w_1}^T\mathbf{x} = \mathbf{w_2}^T\mathbf{x} = 1 \tag{1.4}$$
  - Both set of weights give the same result, but $\mathbf{w_2}$ is preferred by the $L2$ regularizer.
  - $L2$ regularizer likes to spread out the weights (encourages the use of multiple features to make a decision).

- Full Loss Function:

$$L = \frac{1}{N}\sum_i \sum_{j \neq y_i} \max(0, f(\mathbf{x_i}; \boldsymbol{W})_j - f(\mathbf{x_i}; \boldsymbol{W})_{y_i} + \Delta) + \lambda \sum_k \sum_l W_{kl}^2 \tag{1.5}$$

- Loss generally cannot be 0 because of the regularizer
- Biases $\mathbf{b}$ can/should use a different regularizer than the weight matrix $\boldsymbol{W}$
  - Biases could be omitted from regularization
  - Regularizing biases often has negligible effect
- The parameter Δ can be set to 1. Δ interacts with λ and they control the same trade-off.

1.12 Softmax Classifier Overview

- We want to build a Softmax Classifier, discussing the following components:
  - Softmax Function
  - Cross-entropy
  - KL-divergence
  - Numerical Stability
- We want to interpret classifier scores as **probabilities**
- raw scores: $\mathbf{s} = f(\mathbf{x_i}; \boldsymbol{W})$
- want: $P(Y = k | X = \mathbf{x_i})$

1.13 Softmax Function

- raw scores: $\mathbf{s} = f(\mathbf{x_i}; \boldsymbol{W})$
- Softmax function:
  - $\sigma : \mathbb{R}^K \rightarrow \mathbb{R}^K$
  - Input: vector $\mathbf{s}$
  - Output:

$$\sigma(\mathbf{s})_i = \frac{e^{s_i}}{\sum_j e^{s_j}} \tag{1.6}$$

- Example:

$$\mathbf{s} = \begin{pmatrix} s_{cat} \\ s_{car} \\ s_{frog} \end{pmatrix} = \begin{pmatrix} 3.2 \\ 5.1 \\ -1.7 \end{pmatrix} \rightarrow^{exp} \begin{pmatrix} 24.5 \\ 164.0 \\ 0.18 \end{pmatrix} \rightarrow^{normalize} \begin{pmatrix} 0.13 \\ 0.87 \\ 0.00 \end{pmatrix} \tag{1.7}$$

- after applying exp we have unnormalized probabilities ≥ 0.
- at the end we have probabilities summing to 1.

- Softmax in PyTorch

```
torch.nn.Softmax(dim: Optional[int] = None)
torch.nn.LogSoftmax(dim: Optional[int] = None)
```

1.14 Cross-Entropy Loss

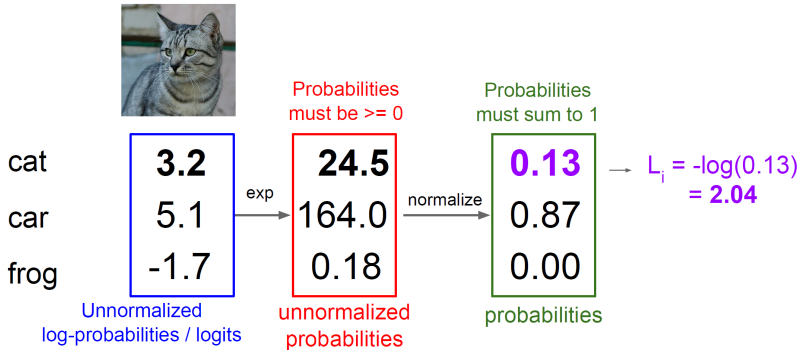
- *Cross-entropy loss* has the form:

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right) \tag{1.8}$$

  - $\mathbf{s}$ is the vector of raw scores
  - y_i is the true class label
- Reformulation:

$$L_i = -s_{y_i} + \log\left(\sum_j e^{s_j}\right) \tag{1.9}$$

- Summary: The Cross-Entropy loss is the negative log of the normalized probability of the correct class
- Example Calculation (cat is the true class):



- Questions about the loss function?
  - What is the min/max possible loss L_i?
  - What is the loss when all raw scores are approx. the same?

1.15 Information Theoretical Interpretation

- **Cross-entropy** between a true distribution P and an estimated distribution Q is defined as:

$$H(P,Q) = -\sum_{x \in \mathcal{X}} P(x) \log(Q(x)) \tag{1.10}$$

  - P and Q are discrete distributions
- Since the true distribution has the form $P = (0, \ldots, 1, \ldots, 0)$ the cross-entropy loss minimizes the cross-entropy defined above.
- Cross-entropy is the sum of **Entropy** and **Kullback-Leibler divergence**

$$H(P,Q) = H(P) + D_{KL}(P \parallel Q) \tag{1.11}$$

- KL-divergence is defined for discrete probability distributions P and Q defined on the

same probability space:

$$D_{\mathrm{KL}}(P \parallel Q) = -\sum_{x \in \mathcal{X}} P(x) \log\left(\frac{Q(x)}{P(x)}\right) \tag{1.12}$$

- Entropy is defined as:

$$H(P) = -\sum_{x \in \mathcal{X}} P(x) \log P(x) \tag{1.13}$$

  - depending on the base of the logarithm the units are bits(2), nats(e), bans(10)
- Note: entropy for the true distribution is 0 therefore the cross-entropy loss is equal to the KL-divergence in our case.

1.16 Probabilistic Interpretation

- Probability of the true class y_i given the image $\mathbf{x_i}$:

$$P\left(y_i|\mathbf{x_i};\boldsymbol{W}\right) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \tag{1.14}$$

  - The cross-entropy loss can be interpreted as minimizing the negative log likelihood of the correct class.
  - This can be interpreted as performing Maximum Likelihood Estimation (MLE)
  - We can now interpret the regularization as a Gaussian prior over $\boldsymbol{w}$ to yield Maximum A Posteriori (MAP) estimation.

1.17 Numerical Stability

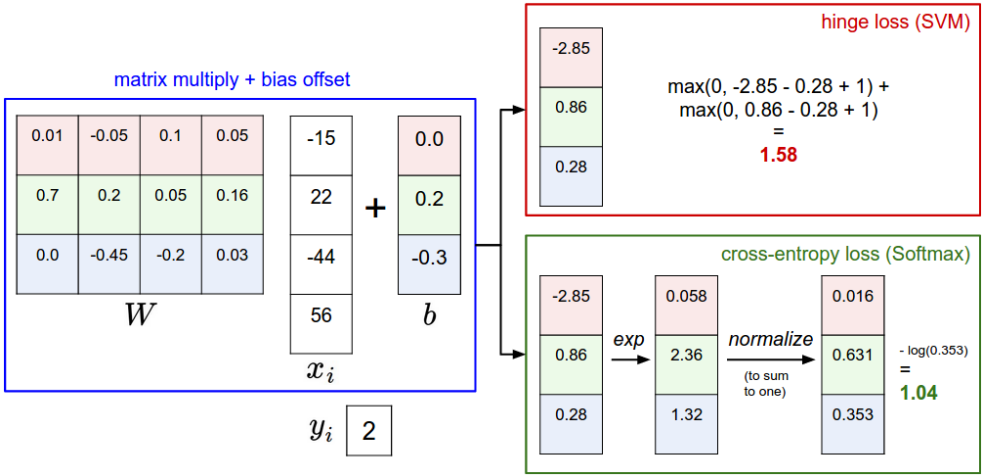
- Computing the softmax can include large values due to the exp function
- We can reformulate:

$$\frac{e^{s_{y_i}}}{\sum_j e^{s_j}} = \frac{Ce^{s_{y_i}}}{C\sum_j e^{s_j}} = \frac{e^{s_{y_i}+\log C}}{\sum_j e^{s_j+\log C}} \tag{1.15}$$

- A good choice is $log(C) = -\max s_j$
  - that means we shift all values in $\mathbf{s}$ so that the max value is 0

1.18 SVM vs. Softmax Comparison

- Example Computation:



1.19 SVM vs. Softmax Comparison

- Softmax is never fully happy / SVM is happy if the margin is satisfied
- Softmax has a better semantic interpretation as probabilities
- Interactive Demo