The elements of classification

D.A. Forsyth,

University of Illinois at Urbana-Champaign

A classifier

- Any procedure that
  - accepts a set of features
  - produces a class
- Hugely useful and general idea
  - Credit card transaction (good/fraud)
  - Should I download this code (secure/insecure)
- Number of classes could be big:
  - Doctor (well, cold, flu, … measles, … etc.)

Summaries of performance

- Error rate:
  - fraction of classification attempts with wrong answer
- Accuracy:
  - fraction of classification attempts with right answer

- Very best error rate
  - Bayes error rate
  - Usually, larger than zero
    - example: alien tries to determine sex from height
  - Usually, very hard to know accurately

Baselines and chance

- Usually, compare performance to baselines
  - other attempts at the classification problem
  - At least, chance:
    - randomly choose a class label for example
    - Two class classifier
      - maximum error rate is 0.5, whatever distribution
    - Multiple classes:
      - depends on the distribution of the classes

Only test error rate matters!

- This makes evaluating a classifier delicate
  - you can't evaluate on training data
    - because the classifier is chosen to do well on that data
      - so estimate is biased optimistic (overfitting)
  - procedure:
    - separate labelled set into disjoint (train set, test set)
    - train on train set
    - evaluate on test set <- this is an unbiased estimate
  - BUT
    - classifier isn't best available (didn't use all data)
    - estimate of accuracy isn't perfect (small data set)

Cross validation

- Iterate:
  - randomly split into (train, test)
  - train on train, evaluate on test
- Accuracy:
  - average of test accuracies in iteration
- Best estimate of classifier:
  - train on all data

Overfitting

- Symptom:
  - good behavior on train, poor behavior on test
- Causes:
  - (usual)
    - classifier is “too flexible”, so fits train well but not test
  - (unusual)
    - training data may not represent general case well
- Common cures:
  - more data
  - simpler classifier

Simple, two-class linear classifier

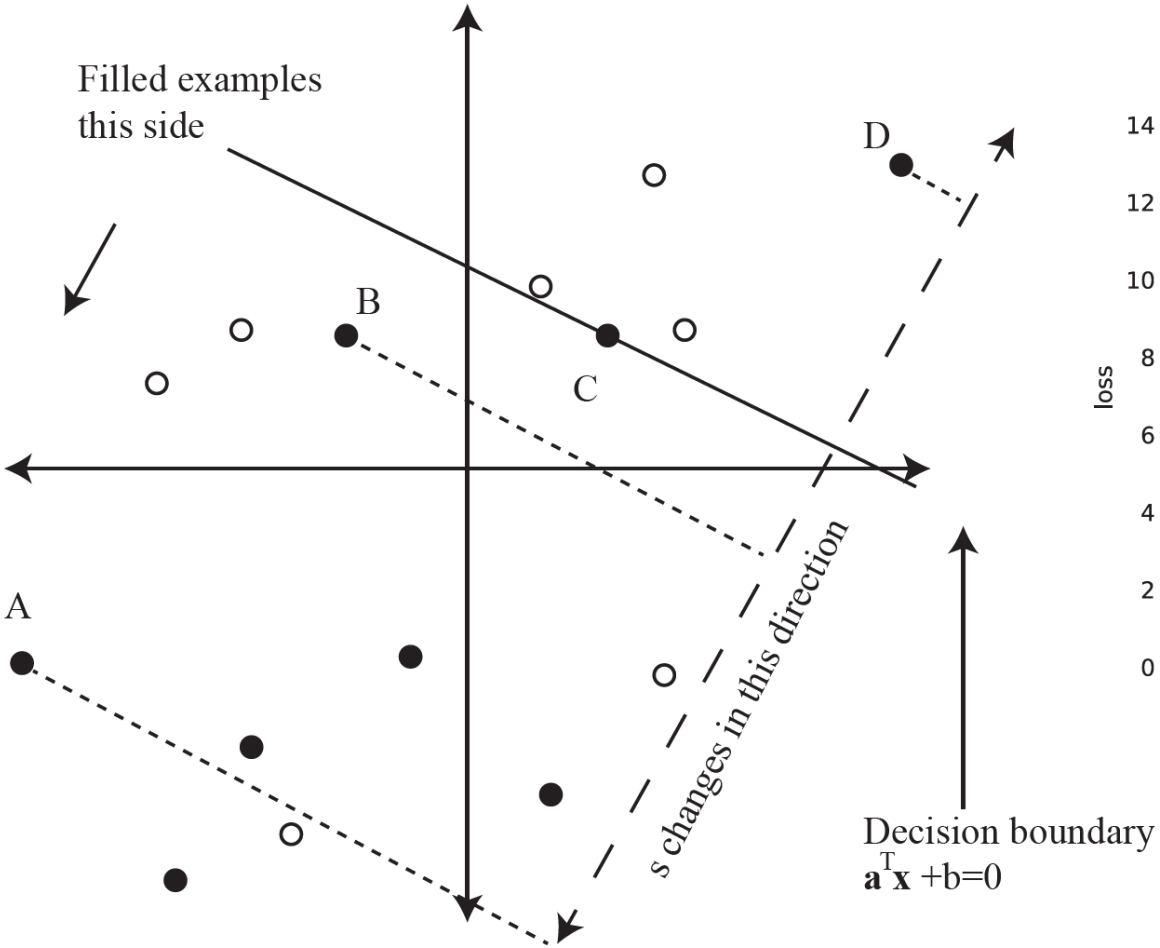
- Assume each data item is (feature vector, label)
  - fv is x

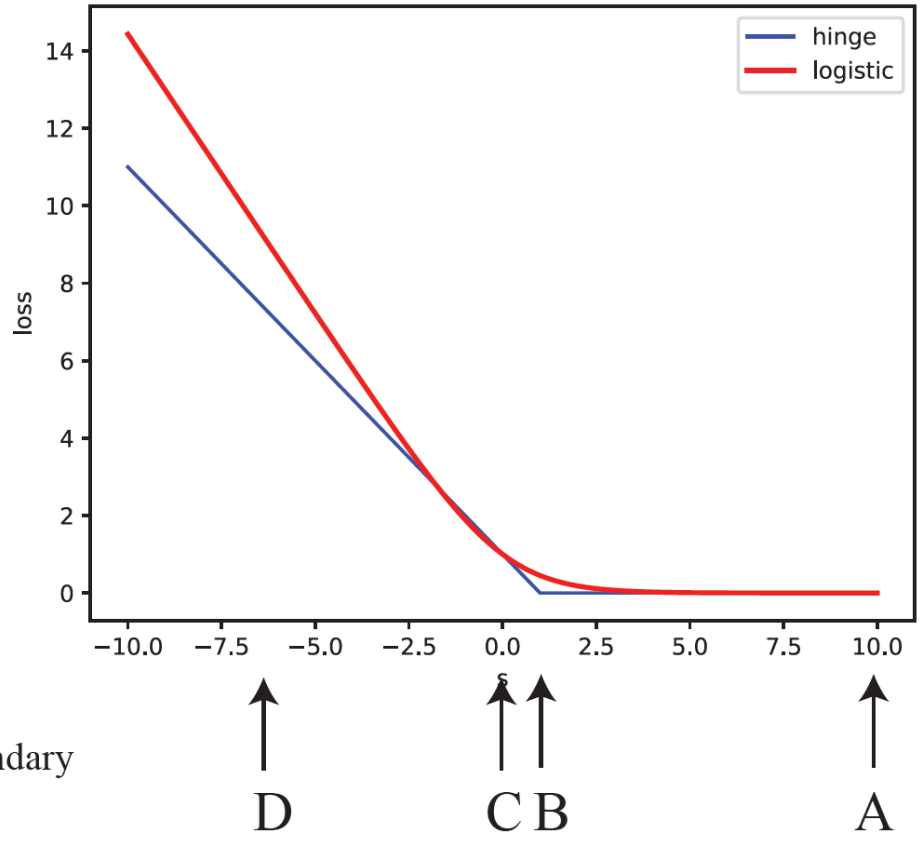
Remember this: *A two-class linear classifier maps a feature vector* $\mathbf{x}$ *to one of two classes, using*

$$sign(\mathbf{a}^T\mathbf{x} + b)$$

where $\mathbf{a}$ *and* b *are parameters of the classifier which are chosen to get strong performance on training data. Linear classifiers are much more powerful than you might expect at first glance. The decision boundary of a linear classifier is a hyperplane in feature space.*

Linear classifier

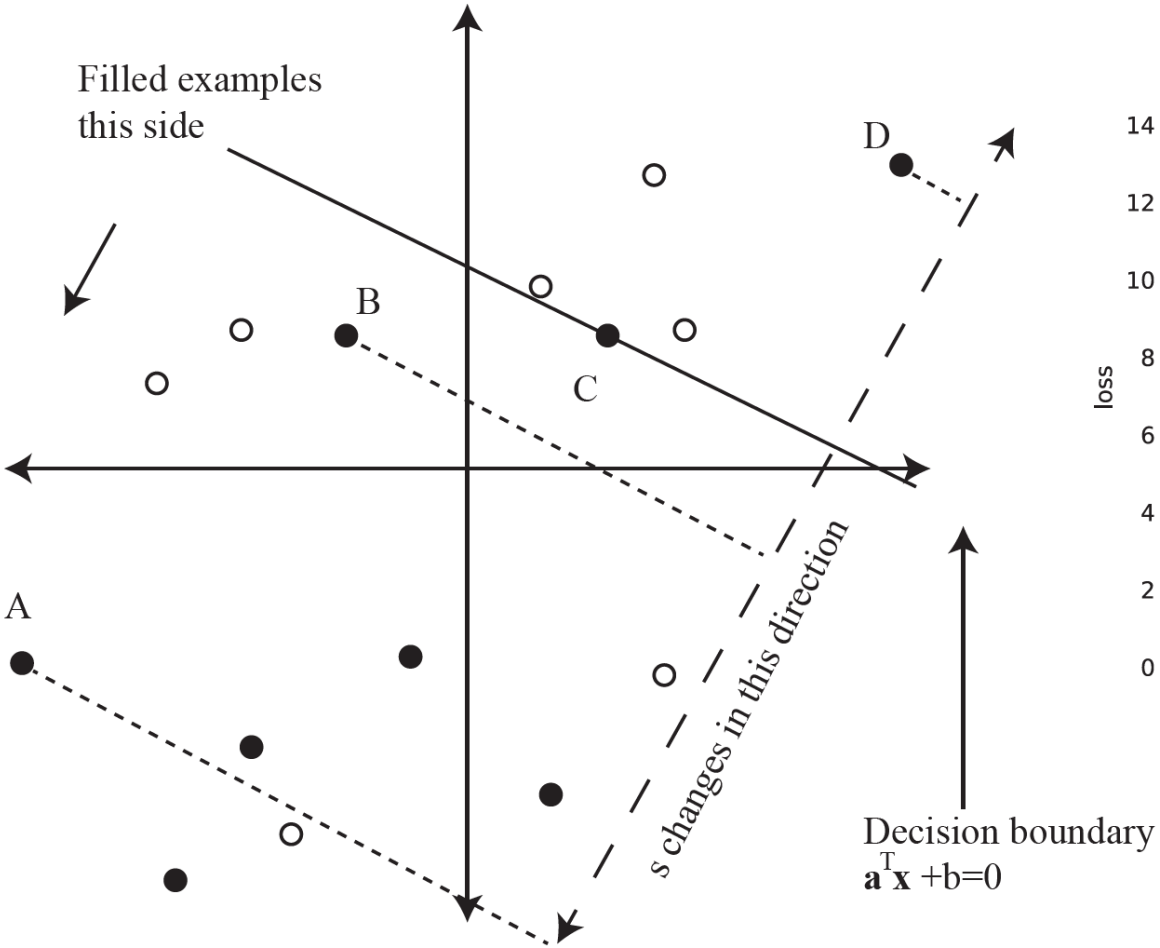


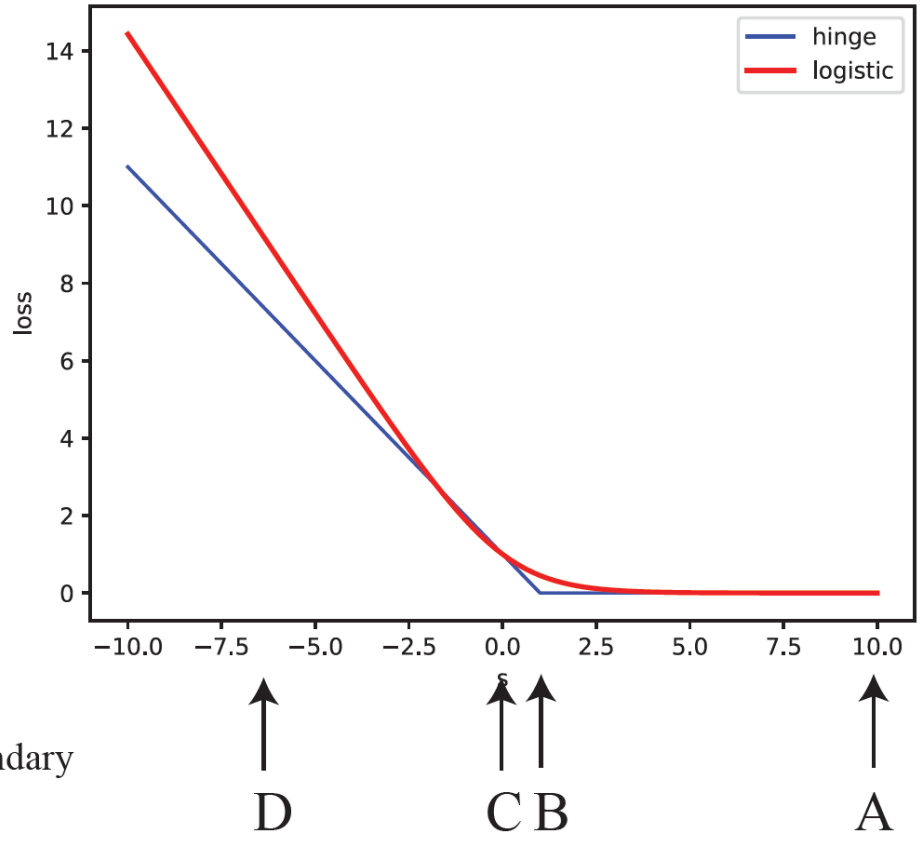


Training a linear classifier

- Choose a, b using training data
  - to minimize some loss
    - error rate doesn't work (unhelpful gradient)
  - losses:
    - logistic regression == cross-entropy == logistic loss
    - hinge loss
  - These are convex approximations to the error rate
- Idea:
  - loss is large if misclassified
  - if correctly classified:
    - small if within margin
    - zero otherwise

Linear classifier





Logistic regression -I

The next step is to choose the parameters of the linear classifier to get good behavior from the classifier. The recipe used in Chapter 17 applies: construct a loss, then use some optimization procedure to minimize the loss. Notice that you can't use training error to adjust the parameters a and b. Gradient descent on error rate won't work, because the gradient is zero almost everywhere. Instead, some approximation to the error rate is required.

One natural approximation is to interpret $u(\mathbf{x}; \mathbf{a}, b)$ in terms of a probability. Use the model

$$u(\mathbf{x}; \mathbf{a}, b) = \log \left[\frac{P(\text{denoise}|\mathbf{x})}{P(\text{real}|\mathbf{x})} \right].$$

This means a data item with positive u is likely to be from the denoiser, and more likely to be from the denoiser if $|u|$ is larger. A data item with a negative u is likely to be real, and more likely so if $|u|$ is larger. In particular

$$P(\text{denoise}|\mathbf{x}) = \frac{e^u}{1+e^u} \text{ and } P(\text{real}|\mathbf{x}) = \frac{1}{1+e^u}.$$

Logistic regression - II

Call this distribution the *predictive distribution* for the i'th example, and write $P(\cdot; u_i)$. Now write $\mathcal{S}$ for the set of N examples, where each example has the form $(\mathbf{x}_i, y_i)$, and

$$y_i = \begin{cases} 1 & \text{if } i\text{'th example is real} \\ -1 & \text{otherwise} \end{cases}$$

Then the log-likelihood of the dataset under this model is

$$\sum_{i \in \mathcal{S}} \left[u_i \left(\frac{1 - y_i}{2} \right) - \log \left(1 + e^{u_i} \right) \right]$$

(you should check this; **exercises**). It would be natural to choose $\mathbf{a}$, b to maximize this likelihood, a procedure known as *logistic regression*. The usual procedure minimizes the negative log-likelihood (also logistic regression). You should think of this as minimizing a loss, and good practice regards a loss as a mean over a set of data items, so write

$$\mathcal{L}_{lr}(\mathbf{a}, b) = \frac{-1}{N} \sum_{i \in \mathcal{S}} \left[u_i \left(\frac{1 - y_i}{2} \right) - \log \left(1 + e^{u_i} \right) \right].$$

Hinge loss

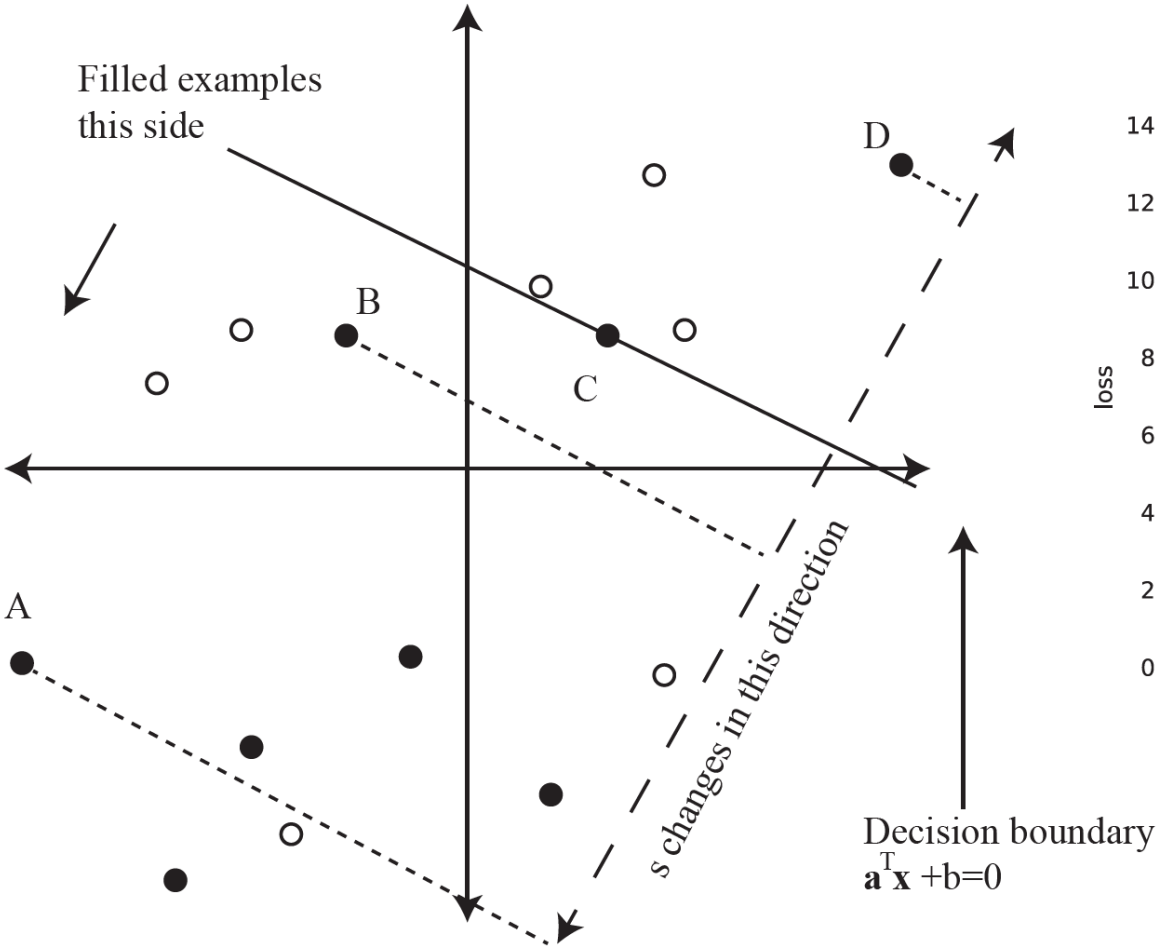
The logistic loss has a helpful geometric interpretation in terms of the hyperplane $\mathbf{a}^T\mathbf{x} + b = 0$. If the i'th example is correctly classified and far from the hyperplane, s_i is large and positive, and so $\mathcal{L}_{\text{logistic}}(s_i)$ is very close to zero. As s_i gets closer to zero (and so the example gets closer to the hyperplane on the right side), the logistic loss grows. If s_i is a lot smaller than zero (and so the example is far from the hyperplane and on the wrong side), the loss grows close to linearly in s_i There are other loss functions that have this behavior. The *hinge loss* function

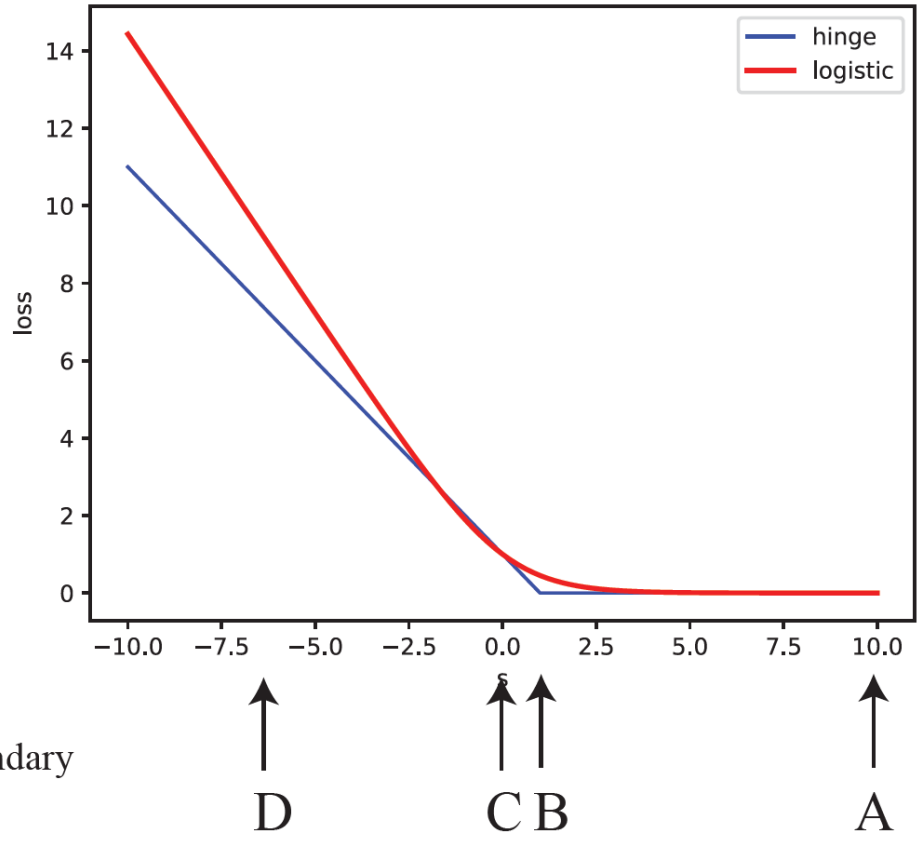
$$\mathcal{L}_{\text{hinge}}(s) = \max(1 - s, 0)$$

has this behavior as well. Recall $s_i = y_i(\mathbf{a}^T\mathbf{x}_i + b)$. The hinge loss for a dataset is

$$\frac{1}{N}\sum_{i \in \mathcal{S}} \mathcal{L}_{\text{hinge}}(s_i).$$

Linear classifier





Where do the features come from?

- Application logic
- Elementary construction
- Existing encoder
- Learn an encoder at the same time as classifier

Practical issue

- Take the block of features from the encoder
  - turn them into a vector of fixed size
  - (this will mean classifier accepts images of fixed size)
- Typically:
  - ensure block shrinks spatially as it moves along encoder
    - by stride or pooling
  - when it is small enough, straighten into vector
- Pooling:
  - make a data block smaller by max/average within fixed size windows (usually 2x2)

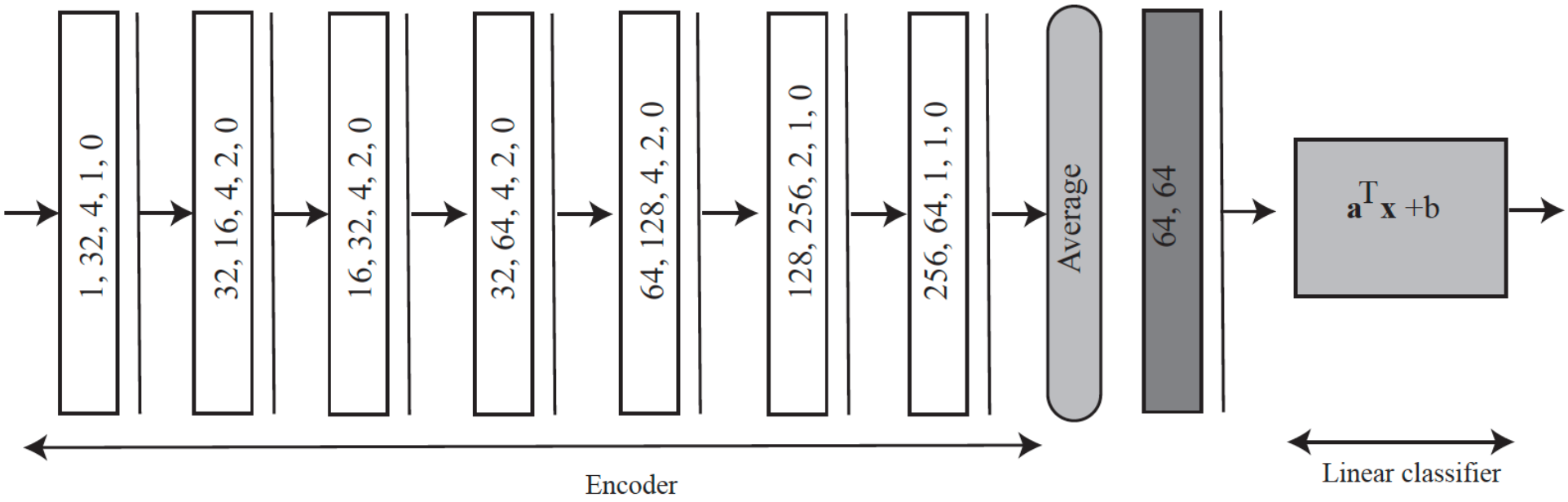
Fully connected layers

- Vector describes the whole image

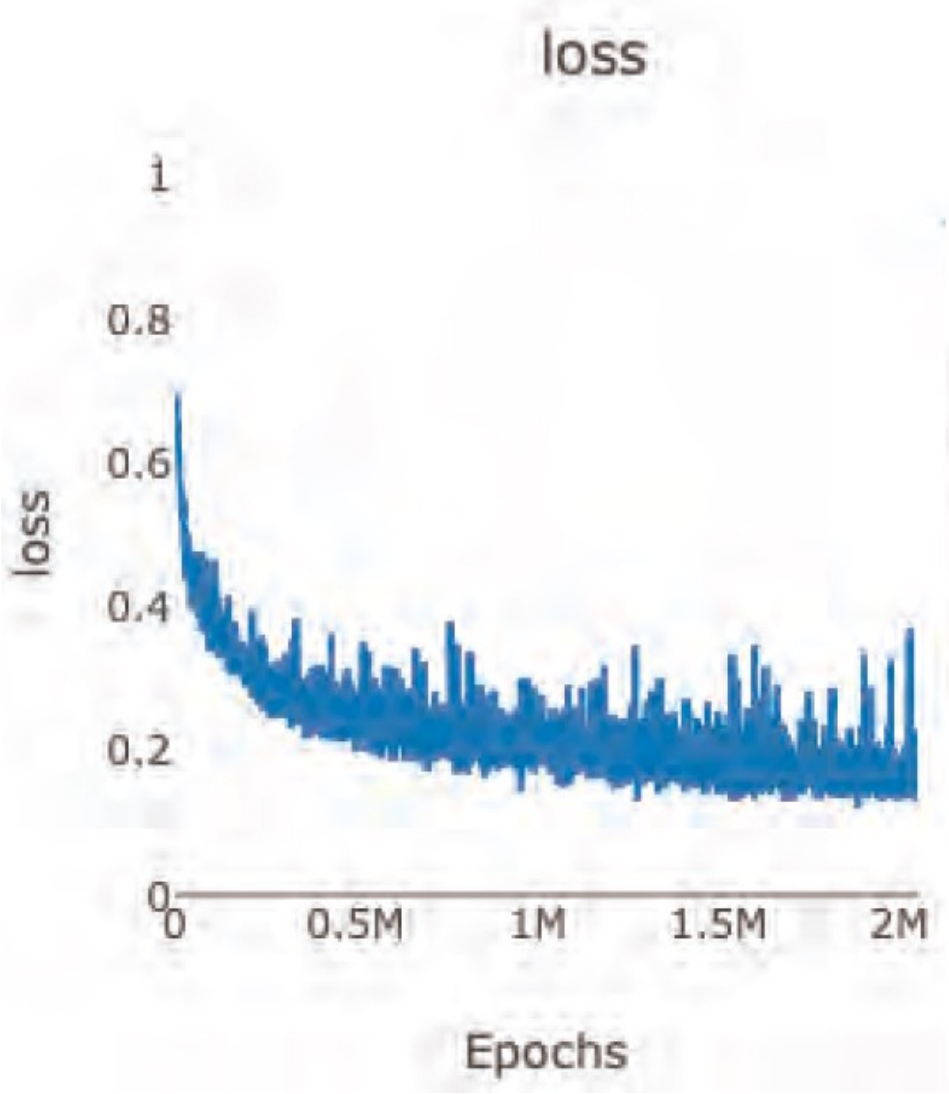
You could regard the $g \times 1 \times 1$ block as a vector (in some APIs, you need to reshape, but this is housekeeping) and simply pass it to a linear classifier. Alternatively, you could transform this vector with a *fully connected layer*, which maps a vector $\mathbf{u}$ to a vector $\mathcal{C}\mathbf{u} + \mathbf{d}$, where the parameters $\mathcal{C}$ and $\mathbf{d}$ are learned, and $\mathcal{C}$ does not need to be square.

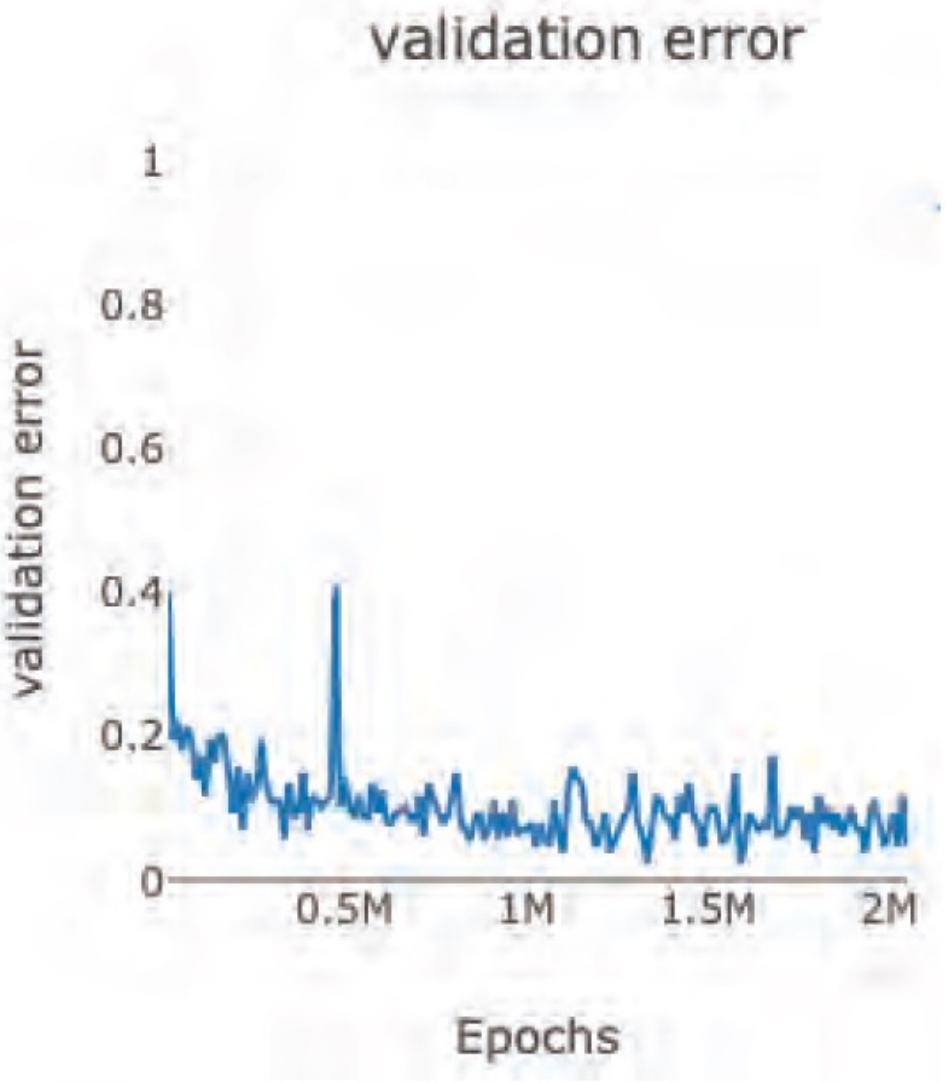
Notice that applying a linear classifier $\mathbf{a}^T\mathbf{x} + b$ to the output of a fully connected layer is not particularly interesting, because the result is $\left(\mathcal{C}^T\mathbf{a}\right)^T \mathbf{u} + \left(\mathbf{a}^T\mathbf{d} + b\right)$, which is just a different linear classifier. Similarly, applying a fully connected layer to another fully connected layer directly is not interesting. Instead, each fully connected layer is followed by a ReLU.

A simple image classifier



Example case: real vs denoised





think about this...

21.1. Check that

$$u(\mathbf{x}; \mathbf{a}, b) = \log \left[\frac{P(\text{denoise}|\mathbf{x})}{P(\text{real}|\mathbf{x})} \right]$$

means

$$P(\text{denoise}|\mathbf{x}) = \frac{e^u}{1+e^u} \text{ and } P(\text{real}|\mathbf{x}) = \frac{1}{1+e^u}.$$

21.2. Now write $\mathcal{S}$ for the set of examples, where each example has the form $(\mathbf{x}_i, y_i)$, and

$$y_i = \begin{cases} 1 & \text{if } i\text{'th example is real} \\ -1 & \text{otherwise} \end{cases}$$

Check the log-likelihood of the dataset under this model is

$$\sum_{i \in \mathcal{S}} \left[u_i \left(\frac{1-y_i}{2} \right) - \log \left(1 + e^{u_i} \right) \right].$$

21.3. Recall

$$H_x(p, q) = -\mathbb{E}[p] [\log q] = -\sum_u p_u \log q_u$$

where the sum is over all elements with non-zero terms in p and q. For fixed p and varying q, check the largest value of $H_x(p, q)$ could be arbitrarily large.

and this...

21.4. Recall

$$H_x(p,q) = -\mathbb{E}[p]\,[\log q] = -\sum_u p_u \log q_u$$

where the sum is over all elements with non-zero terms in p and q. For fixed p and varying q, check the smallest value of $H_x(p,q)$ is $H(p) = -\sum_u p_u \log p_u$. **Hint:** Quick if you remember that $q_u \geq 0$ and $\sum_u q_u = 1$, frustrating if you don't.

21.5. Section 21.2.3 has: "This is convenient when you are not certain of the label, but have some probability that you believe." Explain.

21.6. Under the hinge loss, an example that is correctly classified has no loss if $\mathbf{a}^T\mathbf{x} + b > 1$. You can think of the margin as being given by the gap between the hyperplanes $\mathbf{a}^T\mathbf{x} + b = 0$ and $\mathbf{a}^T\mathbf{x} + b = 0$. What happens to the margin if $\mathbf{a}^T\mathbf{a}$ gets bigger?

21.7. You are given two linear classifiers trained on a large dataset. They have about the same training error, but $\mathbf{a}_1^T\mathbf{a}_1$ is much bigger than $\mathbf{a}_2^T\mathbf{a}_2$. Use the result of the previous example to choose the one you prefer, and explain why.

21.8. How is your choice in the previous exercise related to the weight decay of Section ???