- Course overview

  logistic regression

  stochastic optimization

  Data & parameter tuning

  Deep networks

  Regularization

  Convolutional Networks

  Embeddings

  Recurrent models

- Supervised Classification

  - logistic classfier

linear classfier. $\underset{\text{weights}}{W}X + \underset{\text{bias}}{b} = y \Rightarrow \begin{bmatrix} -2.0 \\ 1.0 \\ 0.1 \end{bmatrix} \xrightarrow{p} \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix}$

trained

$$\underset{\text{Logistic scores}}{y} \Rightarrow \underset{\text{softmax func}}{S(y_i) = \frac{e^{y_i}}{\sum_j y_j}} \Rightarrow \underset{\text{probablities}}{p = 0.7}$$

- One-hot Encoding

$$A = \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix} \begin{matrix} \to a \\ \to b \\ \to c \end{matrix} \Rightarrow a = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad b = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad c = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

- Cross-Entropy

Compare classifier outcomes to one-hot encoding vectors to see how well the classifier does:

$$A = \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix} \longleftrightarrow a \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \; b \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \; c \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Cross-entropy

$$\begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix} \quad D(S, L) = -\sum_i L_i \log(S_i) \quad \begin{bmatrix} 1.0 \\ 0 \\ 0 \end{bmatrix}$$

↑ Distance (entropy)

note: $D(S,L) \neq D(L,S)$

- Multinomial logistic classification

$$\begin{bmatrix} \\ \\ \end{bmatrix} \xrightarrow[WX+b]{\text{linear model}} \begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix} \xrightarrow[S(y)]{\text{softmax}} \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix} \xrightarrow[\text{cross-entropy}]{D(S,L)} \begin{bmatrix} 1.0 \\ 0 \\ 0 \end{bmatrix}$$

↑ X (Input) ↑ Logit $S(y)$ 1-hot labels

- Minimize Cross Entropy

$D(S(Wx+b), L)$

want: low entropy for correct class

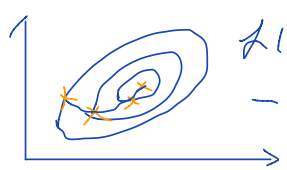
high entropy for incorrect class

Training Loss: average cross-entrophy

$$\mathcal{L} = \frac{1}{N} \sum_i D(S(Wx_i+b), L_i)$$

big sum big matrix

- Gradient Descent

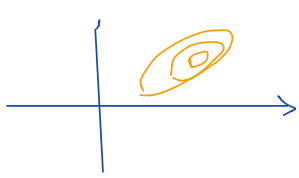


$\mathcal{L}(w_1, w_2)$

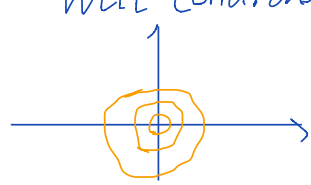
$- \alpha \nabla \mathcal{L}(w_1, w_2)$

- Normalized inputs and initial weights

Badly conditioned



Well conditioned



zero mean, equal variance

Normalization: $\frac{x^{(i)} - \bar{x}}{\bar{x}}$

- Initialization of logistic classfier

$$\mathcal{L} = \frac{1}{N}\sum_i D(S(Wx_i + b_i), L_i)$$

random gaussian distributed weights

- optimization:

$$W \leftarrow W - \alpha \Delta_W \mathcal{L}$$

$$b \leftarrow b - \alpha \Delta_b \mathcal{L}$$

- Measuring Performance

$$\underbrace{\text{Traing} \xleftrightarrow{\text{tune}} \text{Test}} \longrightarrow \text{unseen test} \quad \text{large Error!!!}$$

You are giving your classifier the test set information through your eyes !!!

Solution: validation set

- Overfitting $\longrightarrow$ Data size

The large the data size, the less likelly you will overfit

- Stochastic Gradient Descent (SGD)

logistic regression: very hard to scale

$\mathcal{L} = \sum_i D_i$ All DATA! for gradient descent

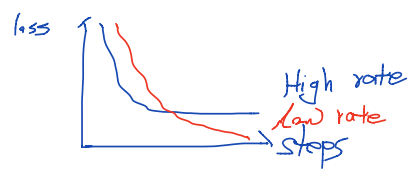
- Helping SGD

Running average: memory of history

Learning rate decay: convergence

- Parameter hyperspace

learning rate tuning:



SGD "Black magic"

— Hyperparameters

- Initial learning rate
- learning rate decay
- momentum
- Batch size
- weigh initialization

- Remember

Keep Calm And
Lower Your
Learning Rate!

- ADGRAD

only need to tune

- Batch size
- Initial weight.