Machine Learning - Deep Learning fundamentals (69152)

Master in Robotics, Graphics and Computer Vision

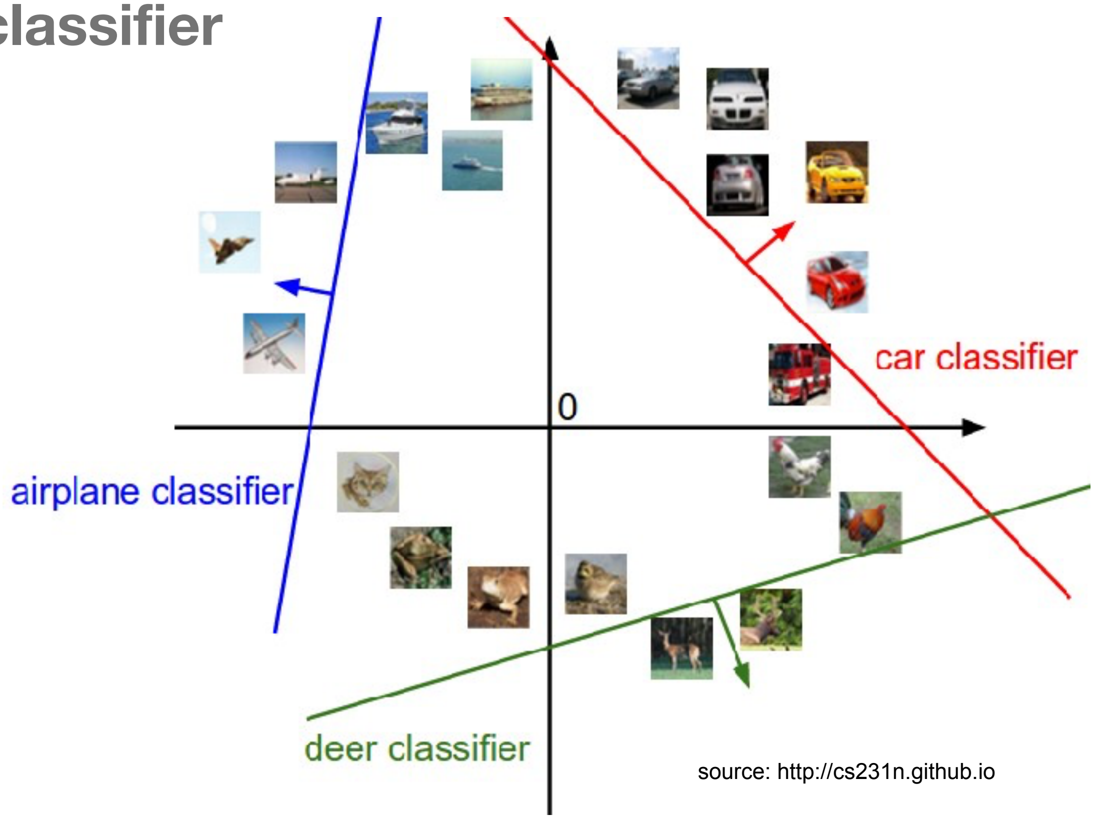
Ana C. Murillo

Universidad Zaragoza

- What's deep learning?
    - Related topics
    - DL pipeline
- **Fundamentals of DL**
    - **Review basic concepts**
    - NN and DNN

Fundamentals of DL

Many basic concepts should be familiar, related to a simple **linear classifier**



source: http://cs231n.github.io

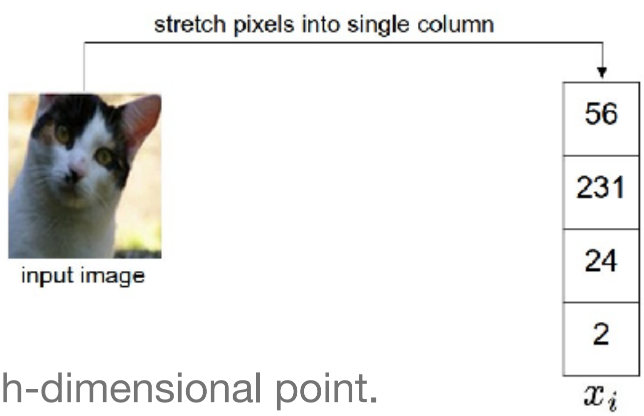
Fundamentals of DL

Score function: map from data to class scores

- **parametric-model**. **w**eights and **b**iases.
  (parametric —> no need to keep all training data)

- linear image classifier example:

$$f(x_i, W, b) = Wx_i + b$$



source: http://cs231n.github.io

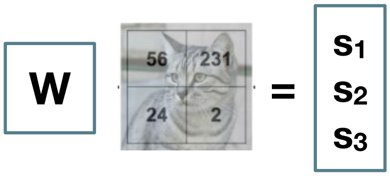
  - image: high-dimensional point.
  - score: weighted sum of all pixel values
  - one classifier (template) per row

What about deep nets?

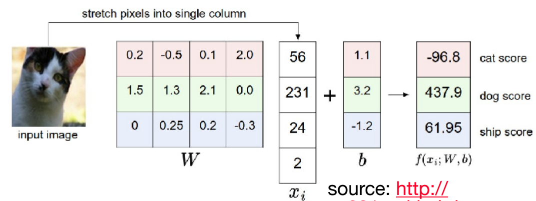
linear classifier

Fundamentals of DL

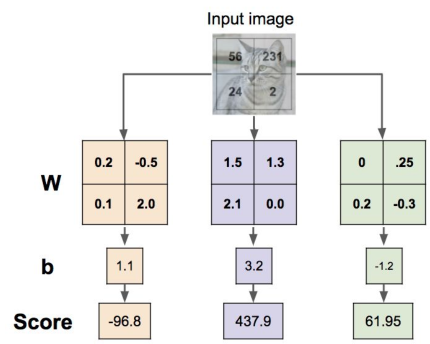
What about deep nets?



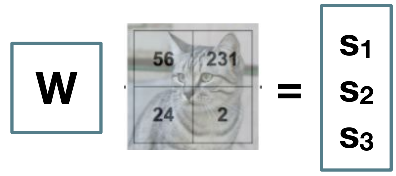
linear classifier



source: http://cs231n.github.io

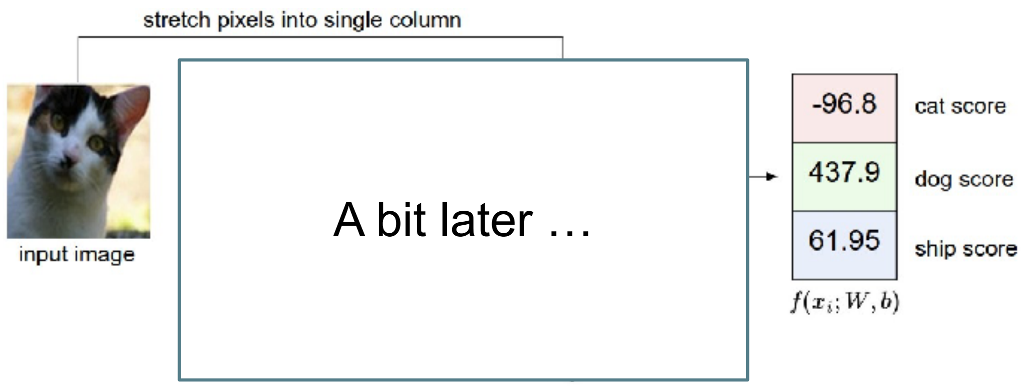


Fundamentals of DL



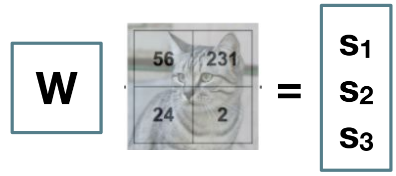
Score function: map from data to class scores

- Deep nets (CNN in particular) also do this with images but much more complex mapping function and params

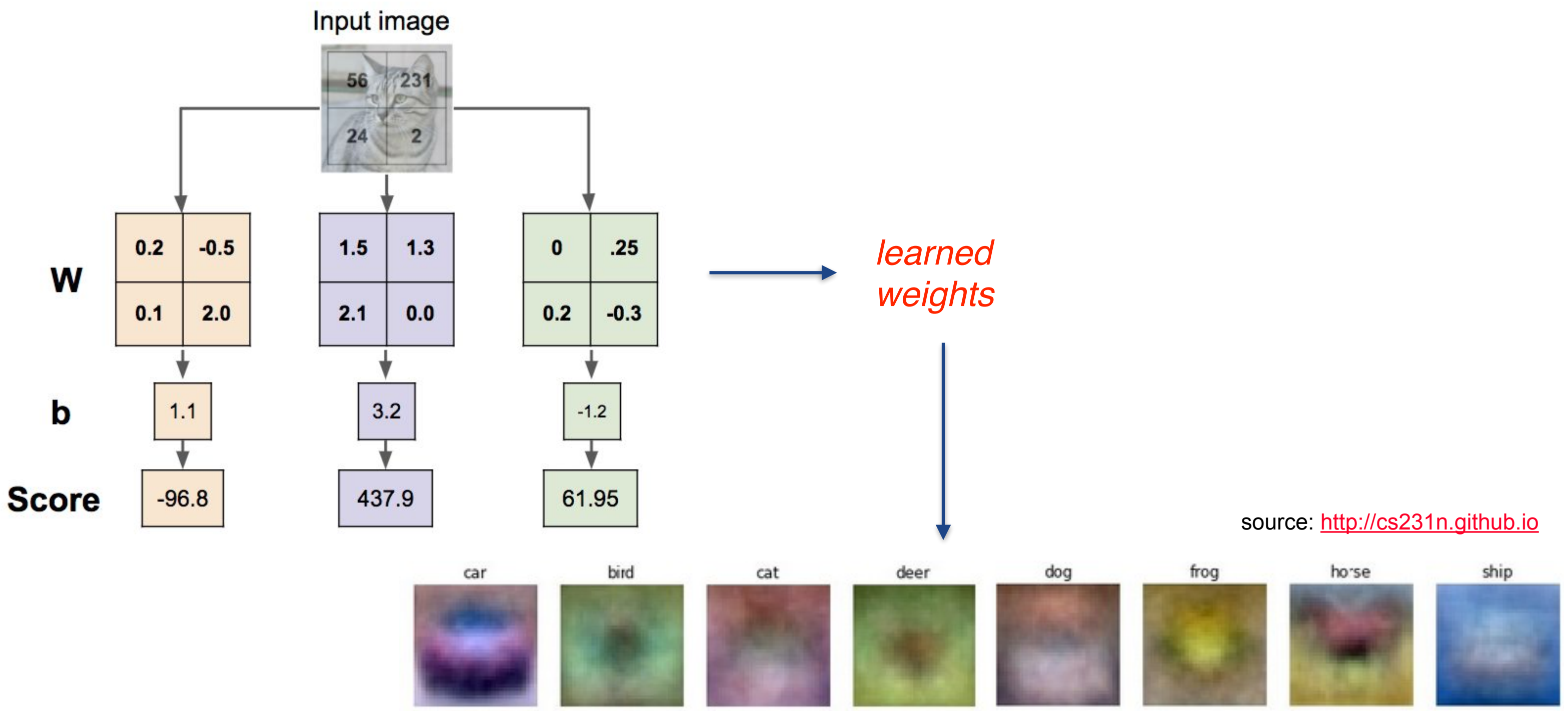


- Common practice:
  - **preprocessing** (e.g., substract mean, normalization)
  - single multiplication $f(x_i, W, b) = Wx_i + b$ $\quad f(x_i, W) = Wx_i$

Fundamentals of DL



- **Score function: map from data to class scores**
  Deep nets (CNN in particular) also do this with images but much more complex mapping function and params (more details in next classes)



How good are the scores?

Loss (cost | objective) function

Fundamentals of DL

Loss function: ***How good are the predictions (scores)?***

$$L = \underbrace{\frac{1}{N}\sum_i L_i}_{\text{data loss}} + \underbrace{\lambda R(W)}_{\text{regularization loss}}$$

Some examples:

- **Multiclass SVM** (one of the possible formulations)
  - *wants* correct class to a have a score higher than incorrect classes by some **fixed margin**

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

the score of every incorrect class - score of actual class

hinge or max-margin loss

Fundamentals of DL

Loss function: ***How good are the predictions (scores)?***

$$L = \underbrace{\frac{1}{N} \sum_i L_i}_{\text{data loss}} + \underbrace{\lambda R(W)}_{\text{regularization loss}}$$

Some examples:

- **Cross-entropy/Softmax loss**:
    - equivalent to "difference" between perfect and actual distribution

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

Softmax Function: score of actual class normalised

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

Cross entropy between distributions p, q

Ideally only the actual label is true: **p = [0, …,1, …,0]**

Fundamentals of DL

Loss function: ***How good are the predictions (scores)?***

$$L = \underbrace{\frac{1}{N}\sum_i L_i}_{\text{data loss}} + \underbrace{\lambda R(W)}_{\text{regularization loss}}$$

"Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error."

(Goodfellow 2016)

- **Regularization**.
  - improve generalization (less overfitting). L2, L1, Dropout, batch-norm, …
  - models preferences: e.g., L2, discourages large values

$$\lambda \sum_k \sum_l W_{k,l}^2$$

regularization ***strength***
(hyperparameter)

$$x = [1, 1, 1, 1]$$
$$w_1 = [1, 0, 0, 0]$$
$$w_2 = [0.25, 0.25, 0.25, 0.25]$$

$$w_1^T x = w_2^T x = 1$$

What's best with L2?
and with L1?

source: http://cs231n.github.io

Fundamentals of DL

Loss function: *How good are the predictions (scores)?*

$$L = \underbrace{\frac{1}{N}\sum_i L_i}_{\text{data loss}} + \underbrace{\lambda R(W)}_{\text{regularization loss}}$$

“Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.”

(Goodfellow 2016)

- **Regularization**.
  - improve generalization (less overfitting). L2, L1, Dropout, batch-norm, …
  - models preferences: e.g., L2, discourages large values

$$\lambda \sum_k \sum_l W_{k,l}^2$$

regularization ***strength***
(hyperparameter)

$$x = [1,1,1,1]$$
$$w_1 = [1,0,0,0]$$
$$w_2 = [0.25,0.25,0.25,0.25]$$

$$w_1^T x = w_2^T x = 1$$

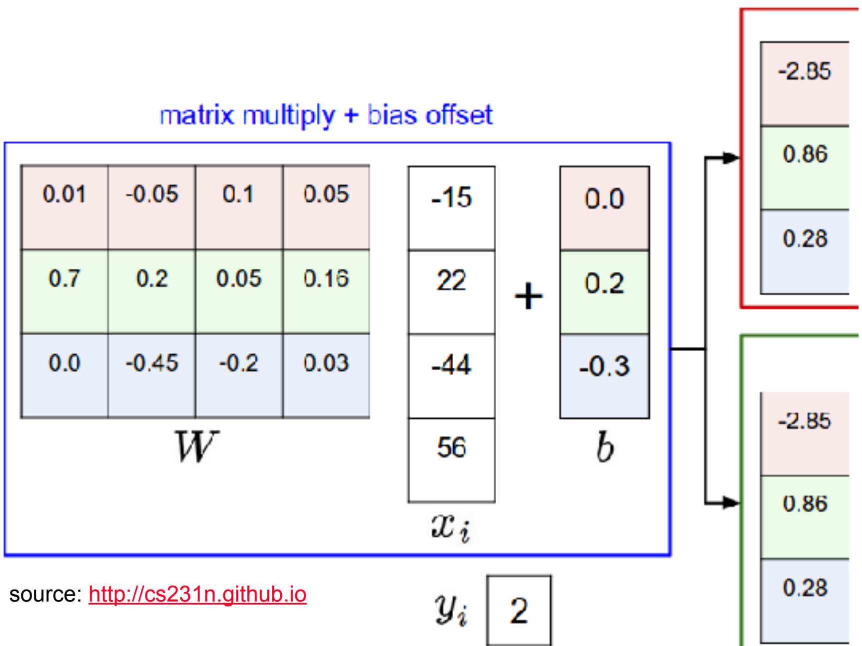
What’s best with L2?
and with L1?

source: http://cs231n.github.io

Loss function - examples

$$L = \underbrace{\frac{1}{N}\sum_i L_i}_{\text{data loss}} + \underbrace{\lambda R(W)}_{\text{regularization loss}}$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$



source: http://cs231n.github.io

Try to compute the two loss functions …

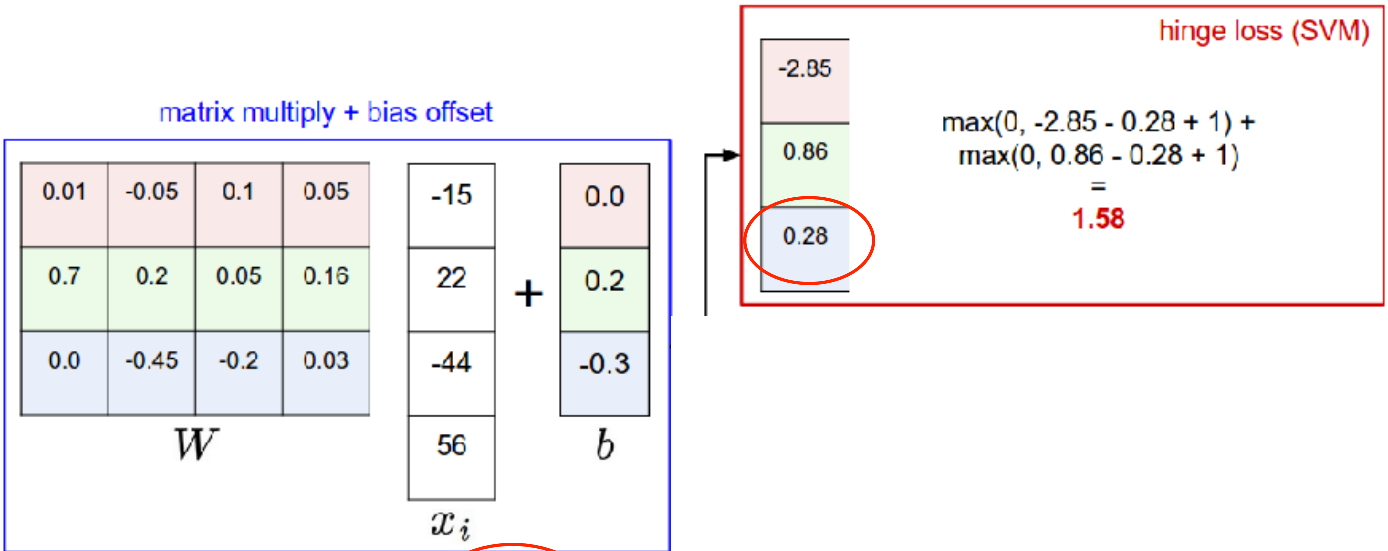
$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

https://b.socrative.com/login/student/

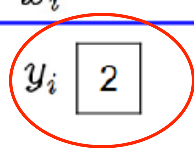
Loss function - examples

$$L = \underbrace{\frac{1}{N}\sum_i L_i}_{\text{data loss}} + \underbrace{\lambda R(W)}_{\text{regularization loss}}$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

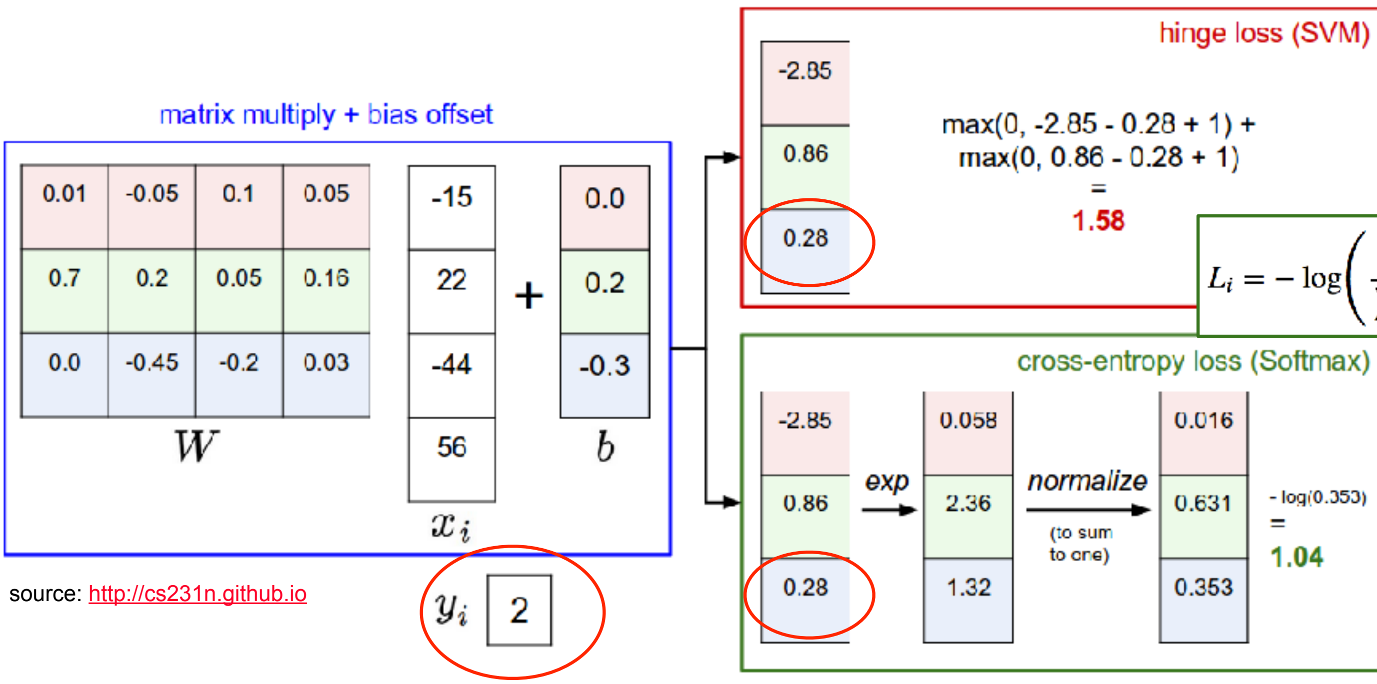


source: http://cs231n.github.io



Loss function - examples

$$L = \underbrace{\frac{1}{N}\sum_i L_i}_{\text{data loss}} + \underbrace{\lambda R(W)}_{\text{regularization loss}}$$



$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

source: http://cs231n.github.io

More examples? http://vision.stanford.edu/teaching/cs231n/linear-classify-demo

Loss function - examples

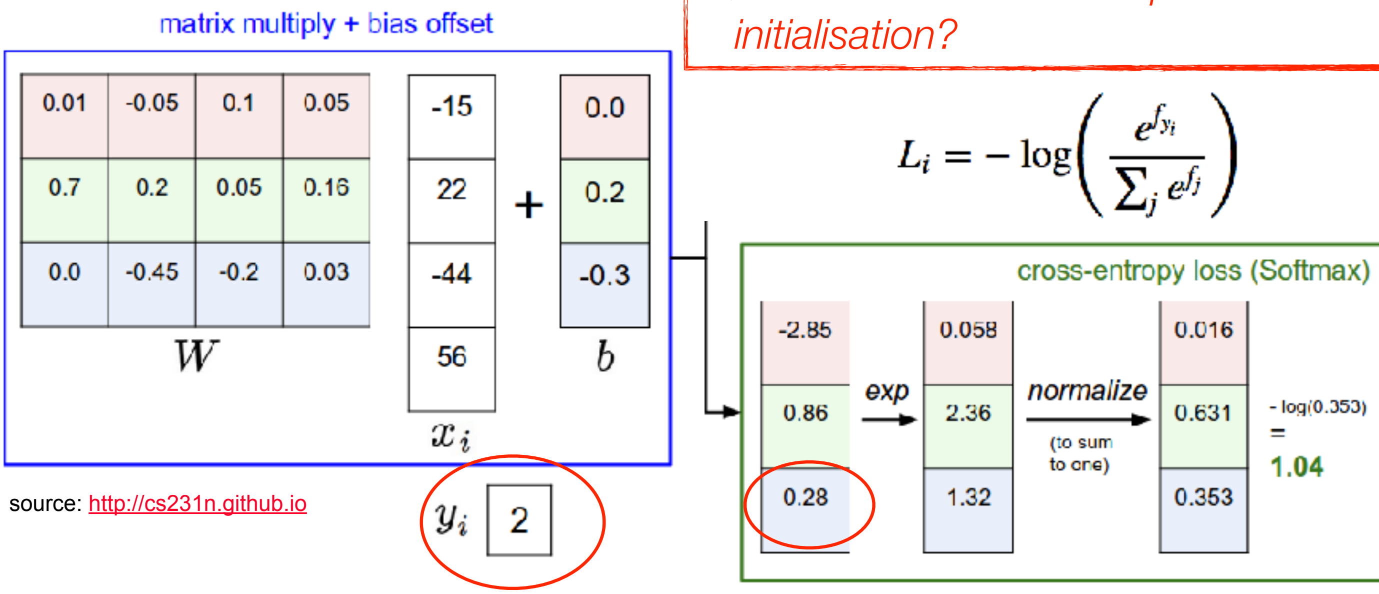
$$L = \underbrace{\frac{1}{N}\sum_i L_i}_{\text{data loss}} + \underbrace{\lambda R(W)}_{\text{regularization loss}}$$

SVM vs Soft-max loss: comparable but different *meaning* of output values

Softmax interprets scores as probabilities

Q1: Can we know the min/max possible L_i?
Q2: What loss can we expect at initialisation?

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$



source: http://cs231n.github.io

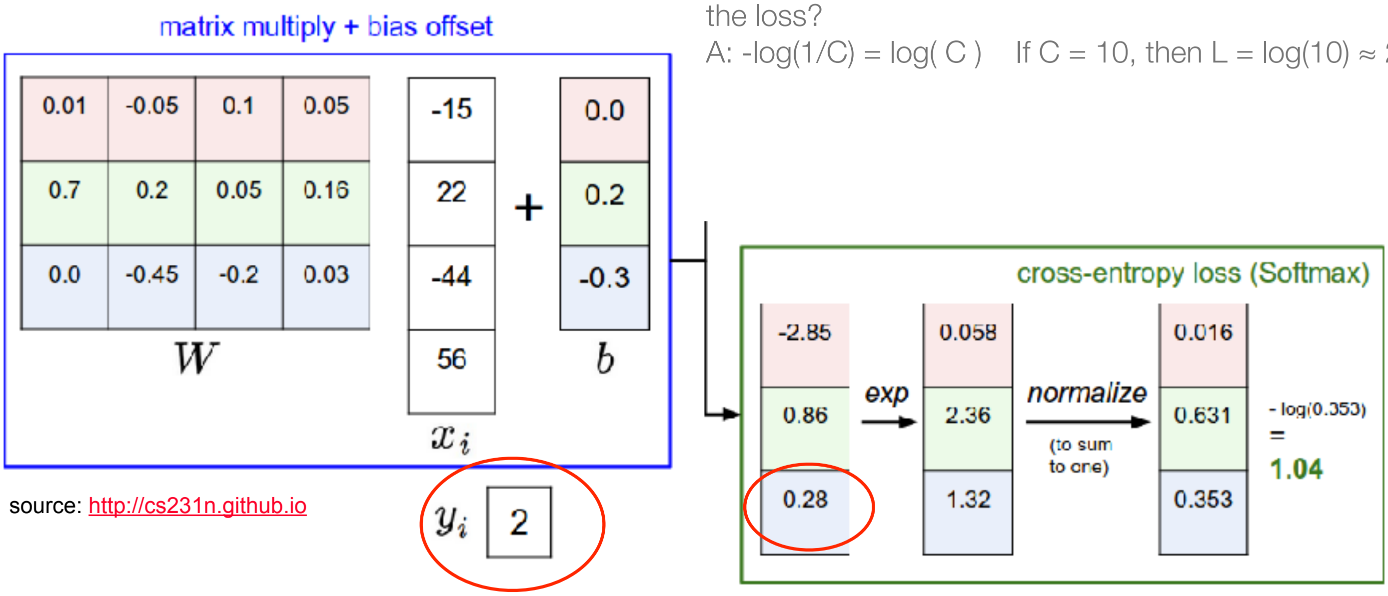
Loss function - examples

$$L = \underbrace{\frac{1}{N}\sum_i L_i}_{\text{data loss}} + \underbrace{\lambda R(W)}_{\text{regularization loss}}$$

SVM vs Soft-max loss:
comparable but different
meaning of output values

Q: What is the min/max possible loss Li?
A: min 0, max infinity

Q2: At initialization all s will be approximately equal; what is the loss?
A: -log(1/C) = log(C)   If C = 10, then L = log(10) ≈ 2.3



source: http://cs231n.github.io

How good are the scores?

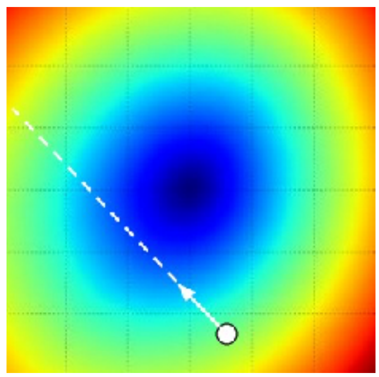
Many options for loss functions

—> critical to choose a right one

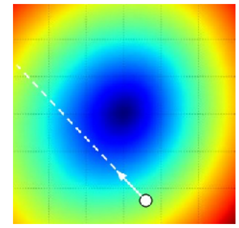
We can measure how good a set of parameters (W) is

How do we find the best W?

Optimisation (training)



Fundamentals of DL



Optimisation —> best way to find the parameters.

We know how to measure how good a set of parameters is.
Start with random weights. Iteratively refine to get lower loss.

Some options …

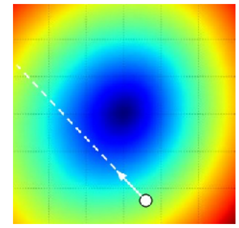
- Random search or “local” search: not *ideal* for DL. Used in other problems
- Follow the “gradient”: vector of derivatives for each dimension of input space
    - compute **gradient of loss function with respect to model weights W**
    - step size (*learning rate*) hyperparameter

$$L(W) = \frac{1}{N} \sum_{i=1}^{N} L_i(x_i, y_i, W) + \lambda R(W)$$

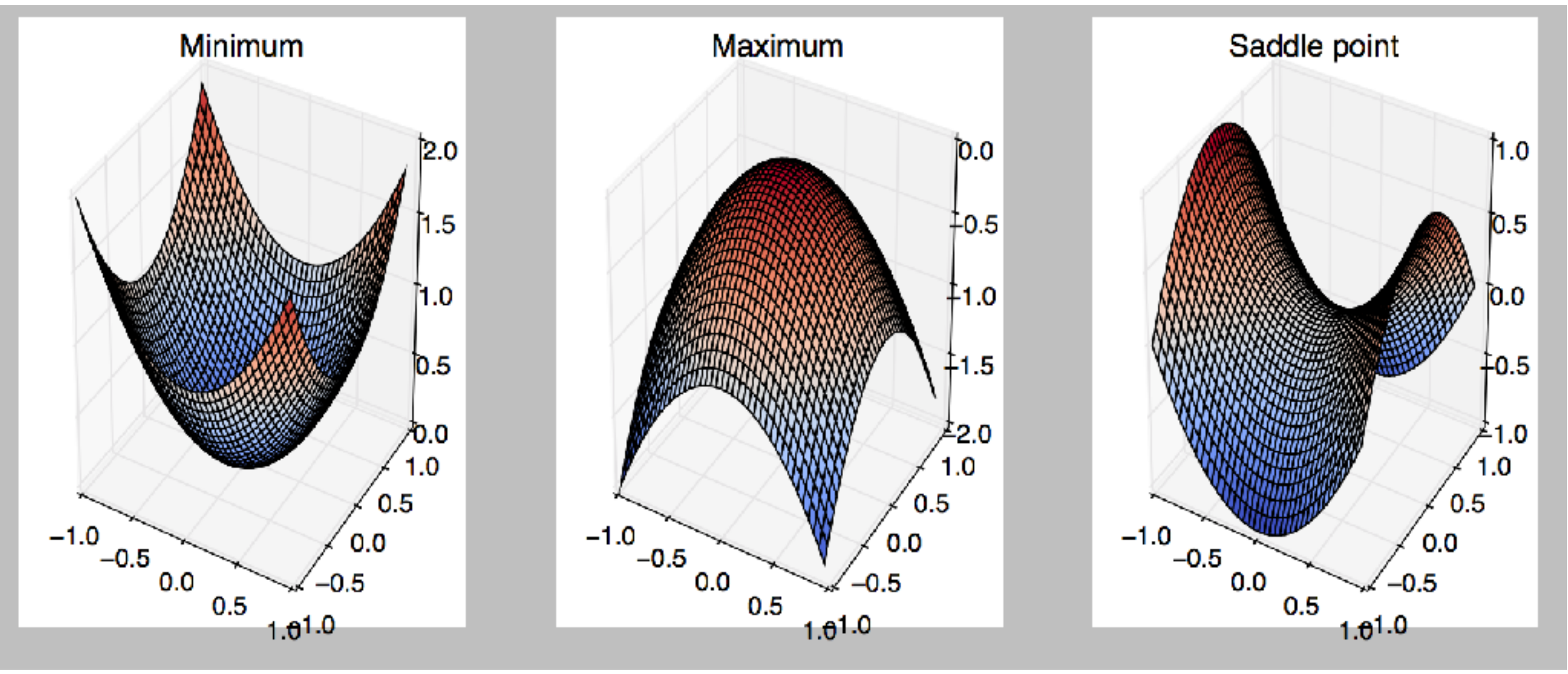
$$\nabla_W L(W) = \frac{1}{N} \sum_{i=1}^{N} \nabla_W L_i(x_i, y_i, W) + \lambda \nabla_W R(W)$$

source: http://cs231n.github.io

Fundamentals of DL



Optimisation. Critical points: Zero gradient, and Hessian with…



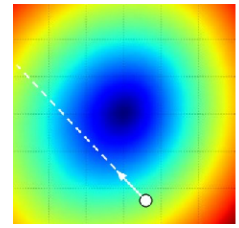
All positive eigenvalues

All negative eigenvalues

Some positive
and some negative

(Goodfellow 2015)

Fundamentals of DL



- **Optimisation** - **GD & Deep Learning**.
    - Ideally, find the exact minimum. Actually, not feasible. N is very large (#samples). W has too many params (1K-1000K)

$$\nabla_W L(W) = \frac{1}{N}\sum_{i=1}^{N} \nabla_W L_i(x_i, y_i, W) + \lambda \nabla_W R(W)$$

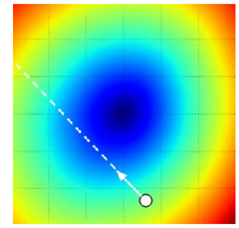
    - Newton method & saddle points **:-(** . SGD has better chances.

$$\theta = \theta - \alpha \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta J_i(\theta)$$ ***GD: for all samples (or BGD)***

$$\theta = \theta - \alpha \nabla_\theta J_i(\theta)$$ ***SGD: for one sample (or a few)***

    - **Stochastic GD:** start "improving" from the first sample processed. One step per sample (or per **mini-batch**: 32, 64, 128, 256, … No more that you can fit in your CPU/GPU memory!).
    - Most common training strategy in DL: SGD + Backpropagation (later)

Fundamentals of DL

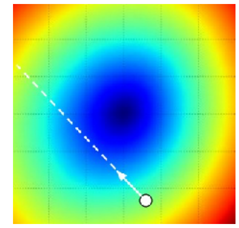


- **Optimisation** - SGD - Still many challenges for the *vainilla* version

¿✅ ❌?	BGD	SGD
1. redundant computations for similar examples		
2. faster		
3. high variance and high fluctuations during training		
4. converges to the minimum of the “basin” where it starts		
5. needs to shuffle training data		
6. suitable for online or incremental learning		

What’s true for which variation?

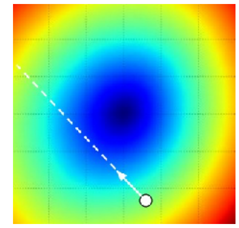
Fundamentals of DL



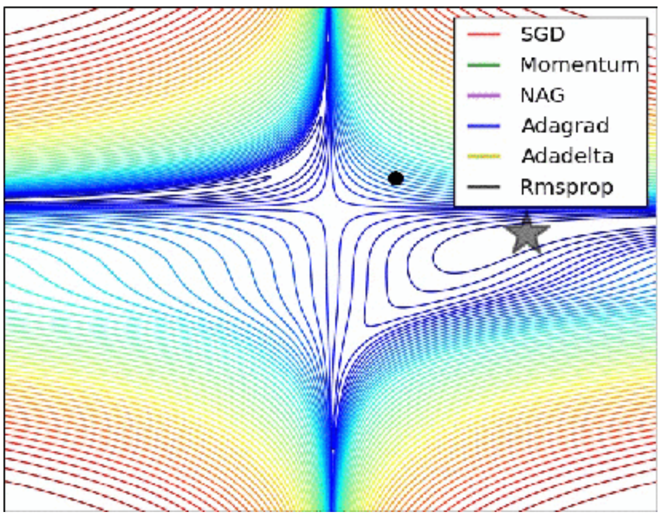
- **Optimisation** - SGD - Still many challenges for the *vainilla* version

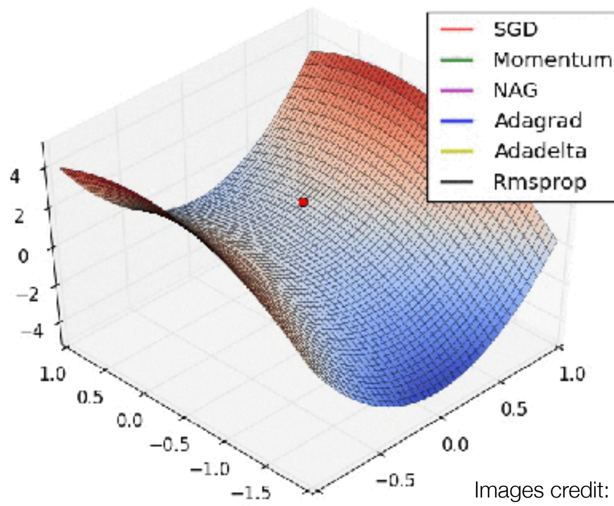
¿✅ ❌?	BGD	SGD
1. redundant computations for similar examples	✅	
2. faster		✅
3. high variance and high fluctuations during training		✅
4. converges to the minimum of the “basin” where it starts	✅	
5. needs to shuffle training data		✅
6. suitable for online or incremental learning		✅

Fundamentals of DL



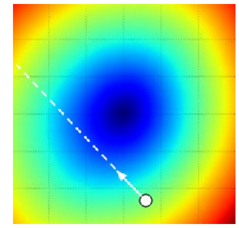
- **Optimisation** - SGD - **Many optimization algorithms** with different properties
  - Contours of a loss surface and time evolution. Different behaviours, speeds, stuck-points, …
  - Saddle point (curvature along different dimension with different signs). Vainilla SGD gets stuck. **Adaptive learning-rate** methods (*Adagrad, Adadelta, RMSprop, Adam, …*) usually get the best compromise in these cases.





Images credit: [Alec Radford](#).

Fundamentals of DL



- **Optimisation** - SGD - Many optimization algorithms with different properties
  - Momentum
  - Average
  - AdaGrad
  - RMSProp
  - Adam
  - Nesterov Accelerated gradient (NAG)

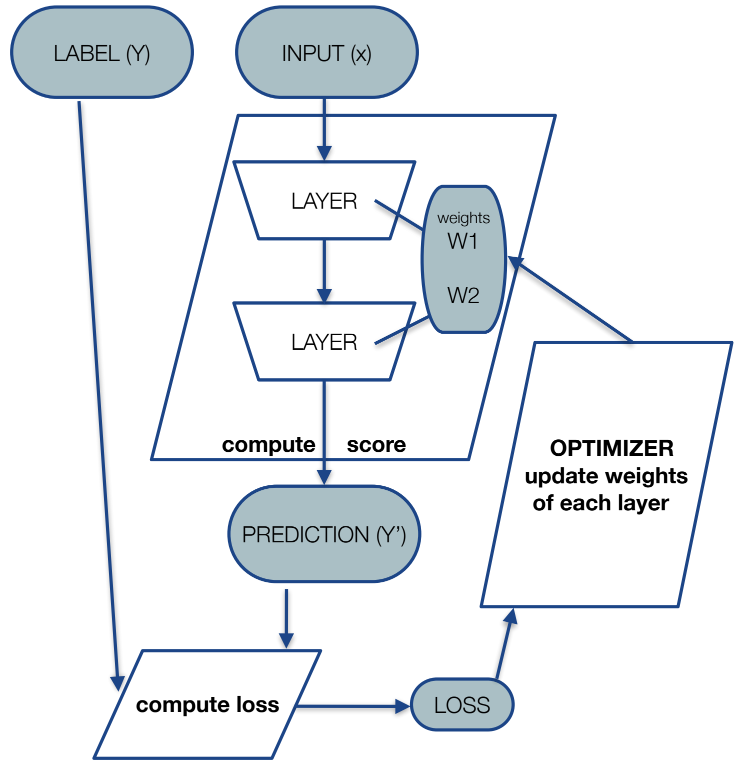
Many available optimizers on standard libraries

Fundamentals of DL - a few key concepts

- **Score** function —> map from data to class scores.
  Weights and biases (w, b)

- **Loss** function —> how good are the predictions (scores).
  SVM, SoftMax, …

- **Optimization** —> best way to find the parameters (w, b): ***Train***



Bibliography - Resources for materials in this block

- Stanford classes on deep learning for Computer Vision (http://cs231n.stanford.edu) and Deep Learning (https://cs230.stanford.edu/)

- Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press, 2016. http://www.deeplearningbook.org

- Deep Learning Summer School Montreal: https://mila.quebec/en/cours/deep-learning-summer-school-2017/