CSE 712 SEM: Seminar on Optimization for Modern Machine Learning

Lecture 1: Introduction

09/01/2022

Instructor: Kaiyi Ji

About Me

- Instructor: Kaiyi Ji, kaiyiji@buffalo.edu
- Office location: Davis Hall 338G
- Course website: *https://cse.buffalo.edu/~kaiyiji/cse712.html*
- Piazza: *piazza.com/buffalo/fall2022/cse712sem/home*
- Office hours: email me for appointments
- Research interests:
  - ✓ Large-scale optimization for big data
  - ✓ Data-driven machine (deep) learning

Course Description

- Prerequisites: CSE 474/574 Introduction to Machine Learning or related courses.

- What we need to do?
  - ✓ Read papers on various optimization methods
    - Loss properties, convex/nonconvex/stochastic/adaptive/minimax/bilevel optimization,
  - ✓ Optimization in machine (deep) learning
    - Applications in linear classification, SVM, neural networks, adversarial learning, GANs, meta-learning, etc.

- Goal of this course
  - ✓ Understand algorithm design and analysis in optimization
  - ✓ Learn how to use optimization in machine (deep) learning
  - ✓ Practice skills of paper reading, presentation, and summary

Logistics

- Location: Davis Hall 113A
- Time: Tuesday 12:00 am to 2:50 pm
- Suggested references:

  - L. Bottou et al. *Optimization methods for large-scale machine learning*. Siam Review, 2018.
  - Y. Nesterov, *"Introductory Lectures on Convex Optimization: A Basic Course,"* Springer, 2004
  - I. Goodfellow, Y. Bengio, A. Courville. *"Deep learning,"* MIT press, 2016.

Logistics

- Each lecture will present a topic covering 1~2 papers
  - Paper(s) listed in **course website (see schedule)**
  - Read paper(s) before each lecture
- Write **one-page** summary after each lecture
  - If there are 2 papers, pick **one** depending on you
  - Due: the day before next lecture

Logistics

- Each student presents one selected paper
  - Sign up a paper by **Today, 11:59 pm**: <Link in course website>
  - Presentation: 30-50 min, 20-40 slides
- Some bonus:
  - Skip summary of your presentation paper
  - Sign up next lecture (9/6): can skip **two more** summary

Logistics

- **Grading policy:**
  - 25% for class participation
  - 45% for writing summaries (15 summaries × 3%)
  - 30% for presentation

Academic Integrity

- Writing paper summaries independently
  - Do not copy others's work or solution
  - Any plagiarism will result in a F score
- Any reference used in your presentation must be cited
  - Online resources: authors' slides
- Academic integrity policies can be found at
  - *https://engineering.buffalo.edu/computer-science-engineering/informati on-for-students/academics/academic-integrity.html.*

Today's Plan

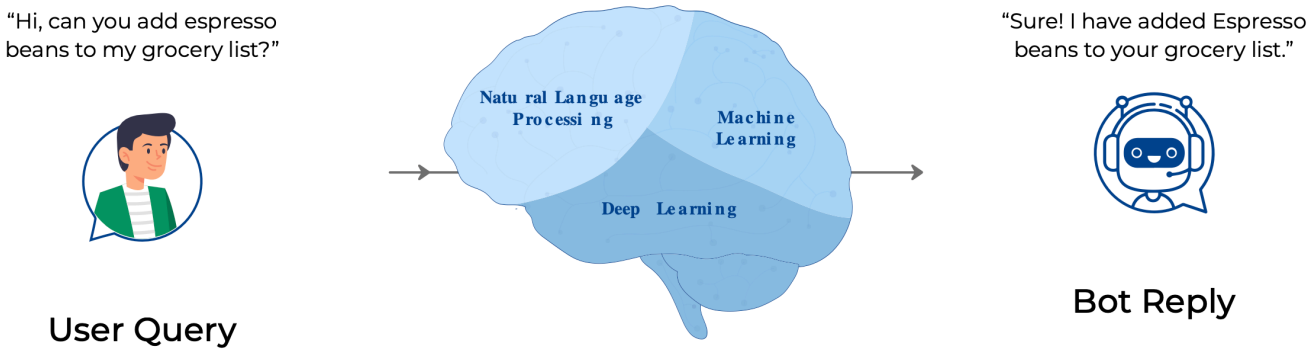
- A short introduction to OptML and today's readings:
  - Nestrov. *Introductory Lectures on Convex Programming.* *Section 2.1.1 and 2.1.5*

    https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.693.855&rep=rep1&type=pdf
  - Bottou et al. *Optimization Methods for Large-Scale Machine Learning.* *Section 2.1 and 2.2*
  - **Optional reading**: Sutskever, Martens et al. *"On the importance of initialization and momentum in deep learning," 2013*

- Submit your summary before next Monday (9/5, 11:59 pm)
  - Under assignment/summary_1 folder in Piazza
  - Please send a **private** message and attach your **summary pdf**

Machine Learning

CV: objective detection



NLP: conversational user interface

Source: GenieTalk.ai

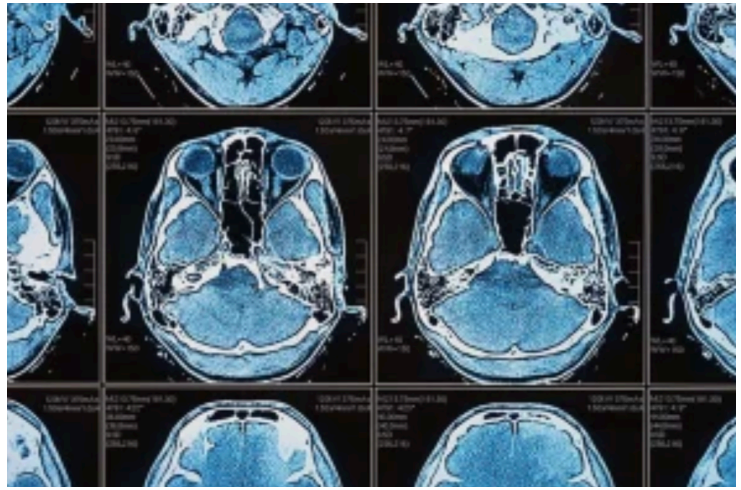


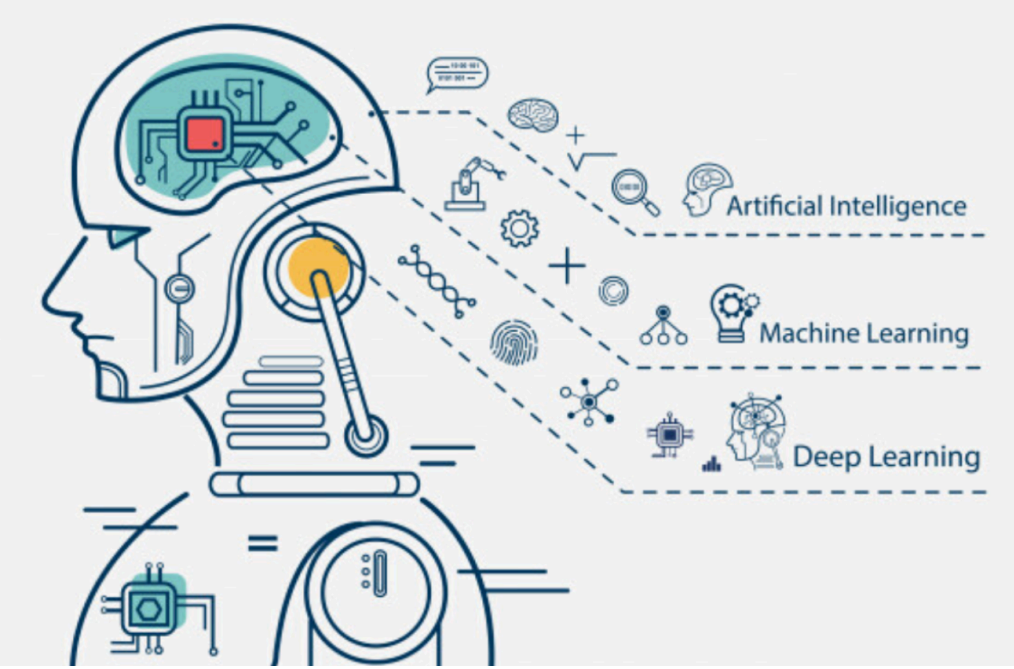
Image processing: disease detection

DL in autonomous driving

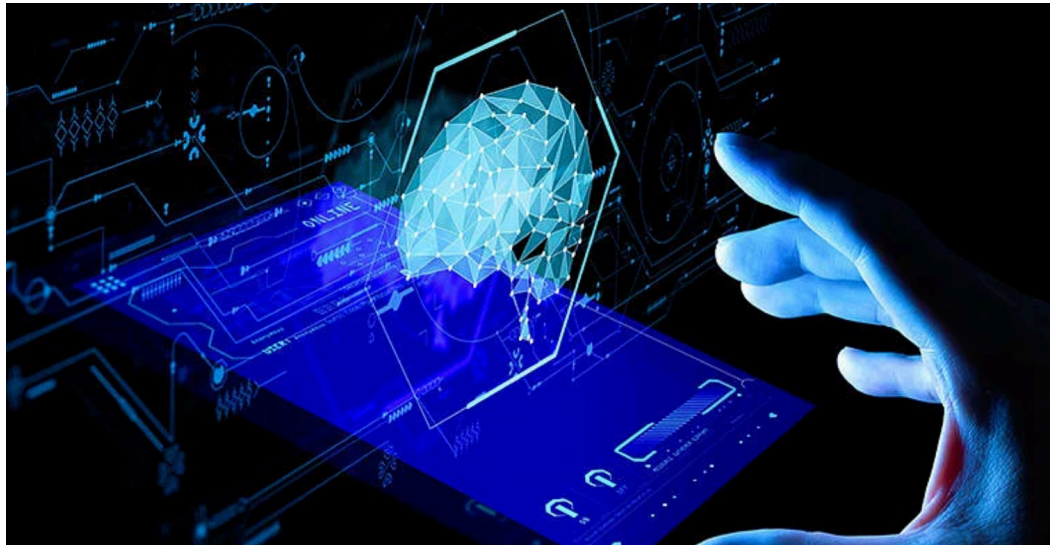
Source: Tian's blog

Engine in Machine Learning

- How to train machine?



Source: nearlearn.com

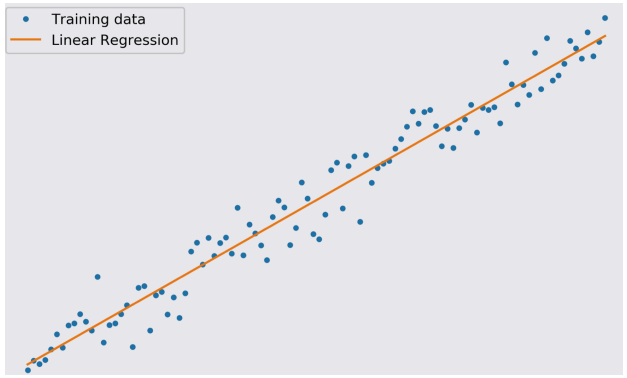


Source: optimization by TensorFlow, Loon's blog

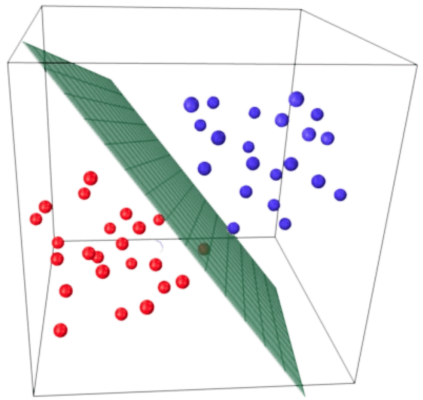
- **Optimization** is **engine** to train intelligence of machine!

Convex Optimization in ML

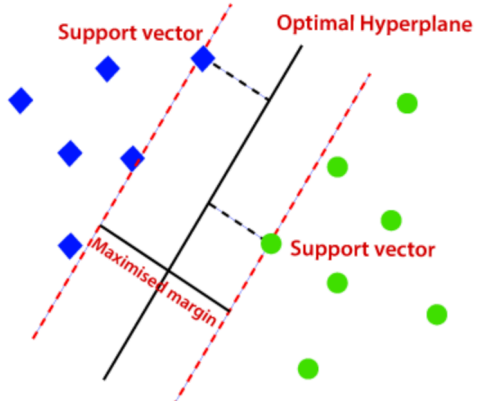
- Linear regression and classification
  - Data sample based formulation
- Support vector machine (SVM)
  - Margin based classification, **strong theoretical guarantee**



Linear regression



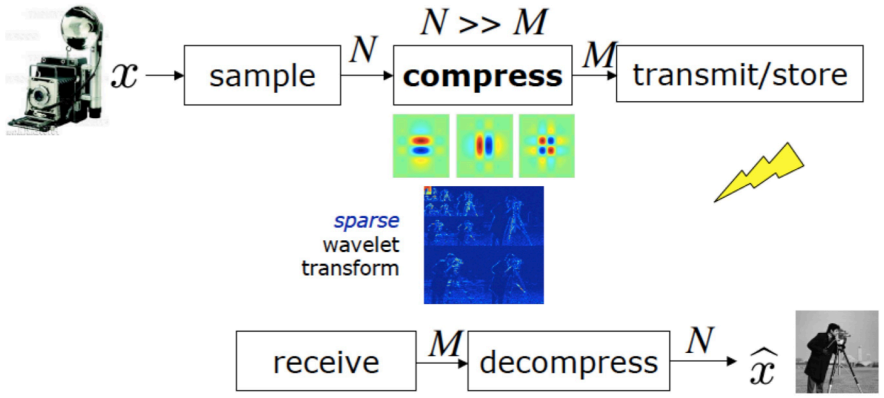
Linear classification



SVM in classification

Structured Optimization in ML

- Compressed sensing
  - Signal compressed, sparse representation
- Matrix completion
  - Recommendation systems, medical study, etc.

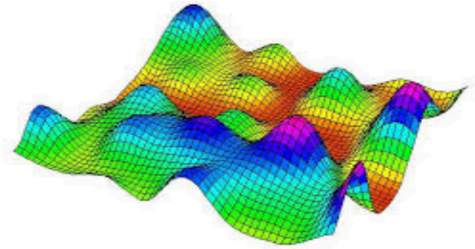


Signal compression in transmission

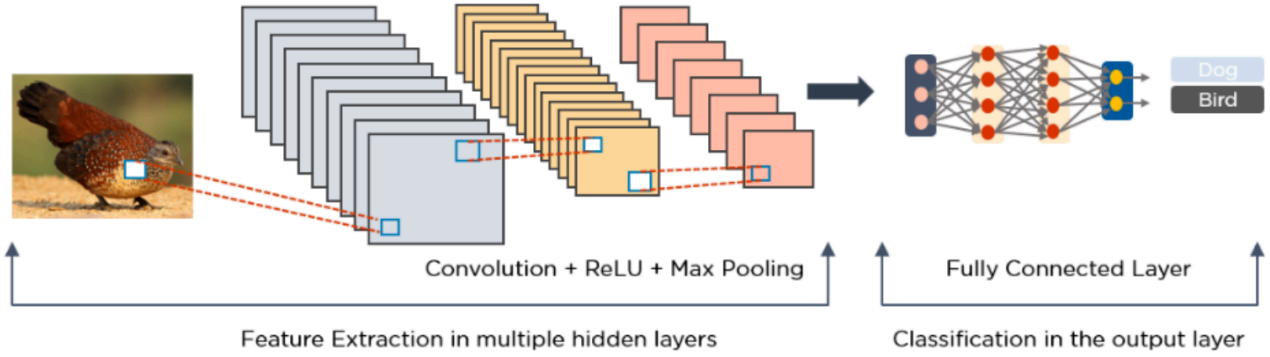
Movie recommendation for users

Resource: V. Morgenshtern's course on matrix completion

Nonconvex Optimization in ML

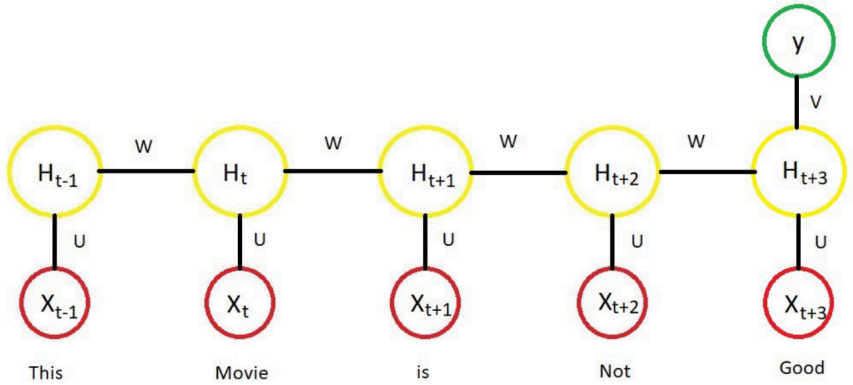


- Training with deep neural networks (highly nonconvex landscape)
  - Multi-layer perceptron (MLP) in classification/regression
  - Convolutional neural networks (CNN) in image processing, vision, etc
  - Recurrent neural networks (RNN), transformer in natural language processing (NLP)



CNN for image processing

Resource: Avijeet Biswal's lesson

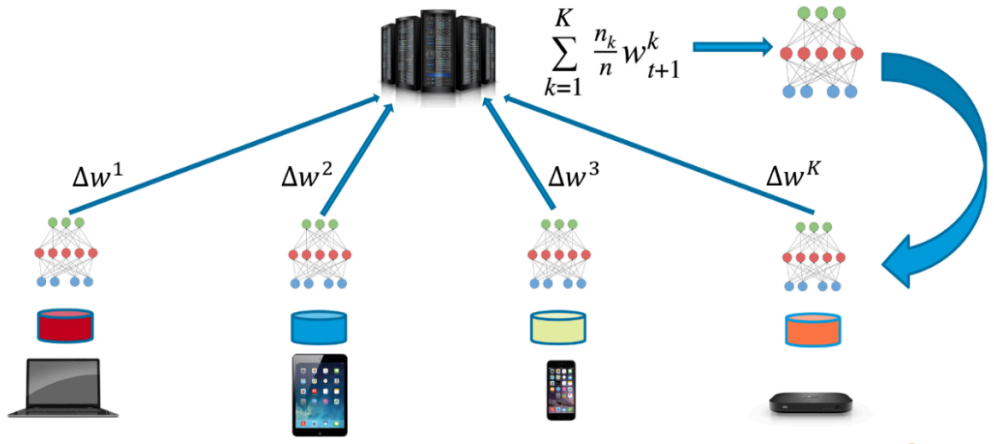


LSTM in NLP

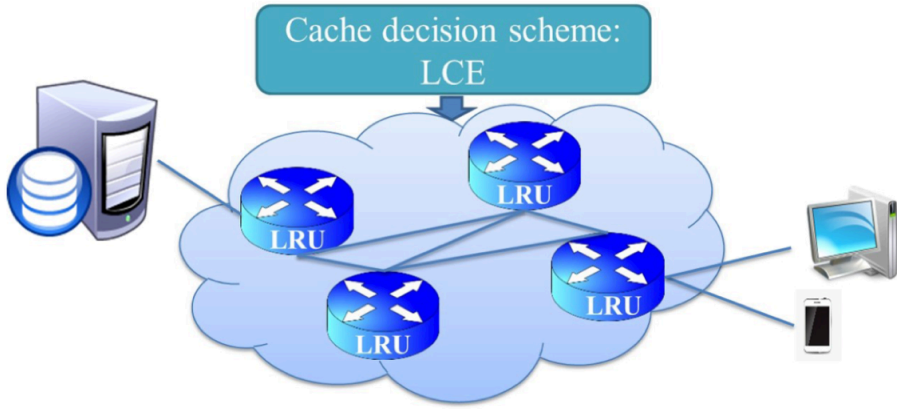
Resource: Vibhor Nigam's blog

Distributed Optimization in ML

- Decentralized/federated/distributed learning over networks
  - Improve scalability over big data and huge models



Federated learning with edge devices



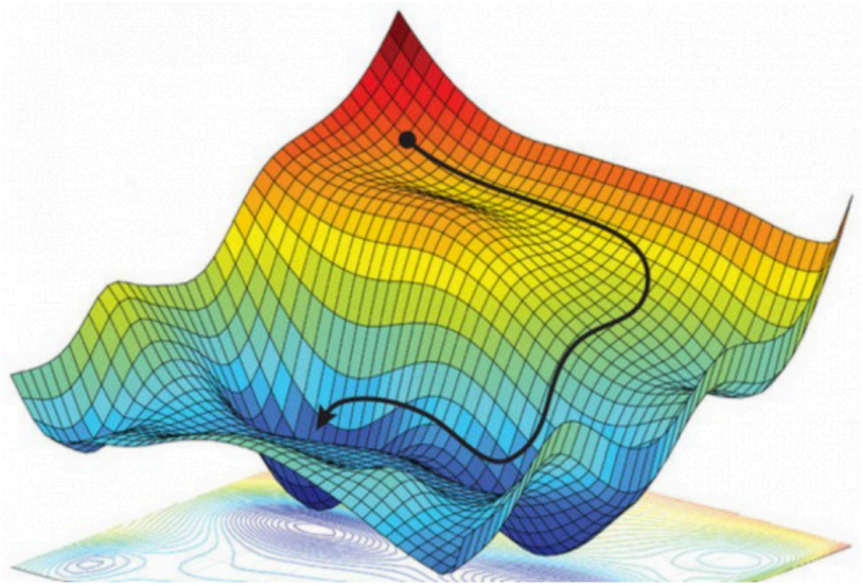
Distributed protocol in Internet!

Today's Focus

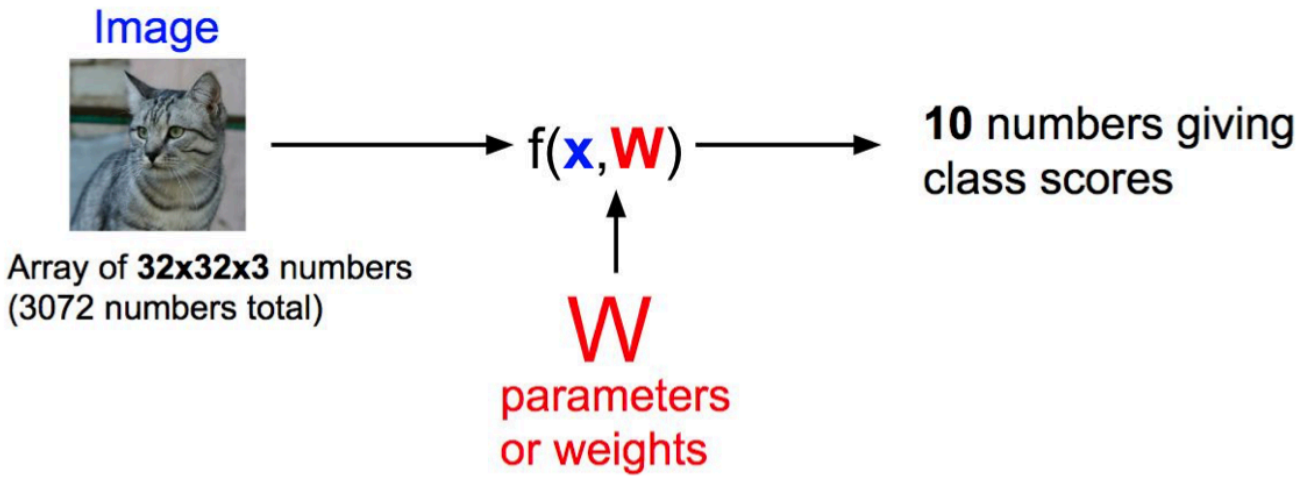
- Linear classification, SVM, logistic regression, etc.
  - Loss functions and optimization

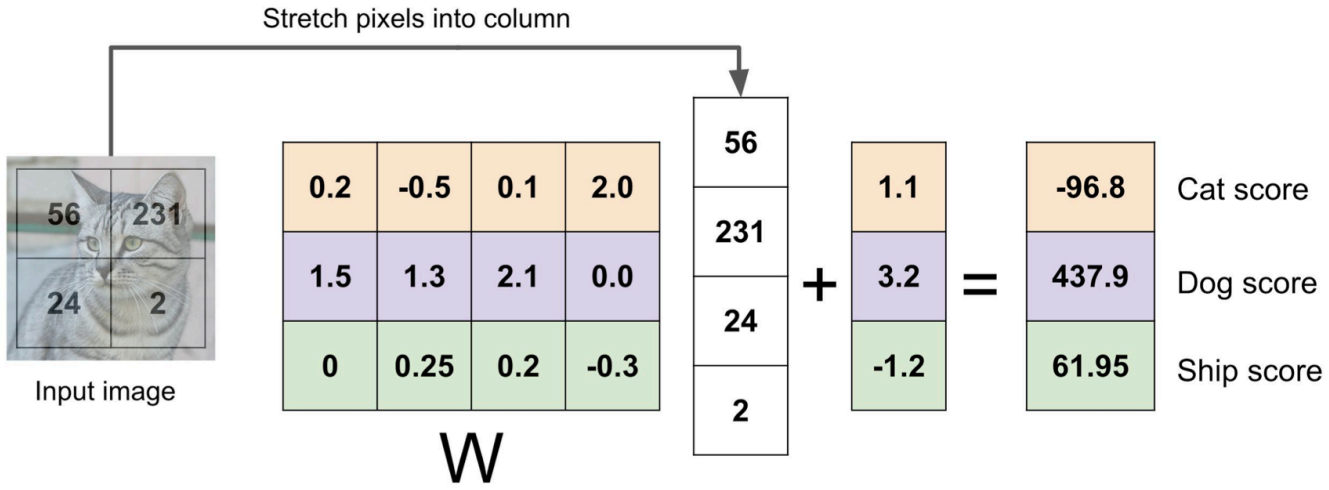
Algorithm foundation

- Convex optimization
  - Properties of convex functions
  - (Stochastic) gradient methods
  - Convergence to global minimum



Linear classification and SVM





Example from stanford cs231n course

- Linear classifier

$$f(x, W) = Wx + b$$

- Three-class example
  - Input: a training sample (image)
  - Output: 3 class scores
  - Current scores:

We do not expect!

Loss function for SVM

3 training samples, 3 classes

Scores under some W: $f(x, W) = Wx \in \mathbb{R}^3$

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

Illustrating example from stanford cs231n course

- Given a dataset of samples $\{(x_i, y_i)\}_{i=1}^{N}$
  - x_i : feature vector (image)
  - y_i : (integer) label (e.g., class number)
- Loss over a data example (x_i, y_i):
  - Score: $s = f(x_i, W)$
  - Hinge Loss:

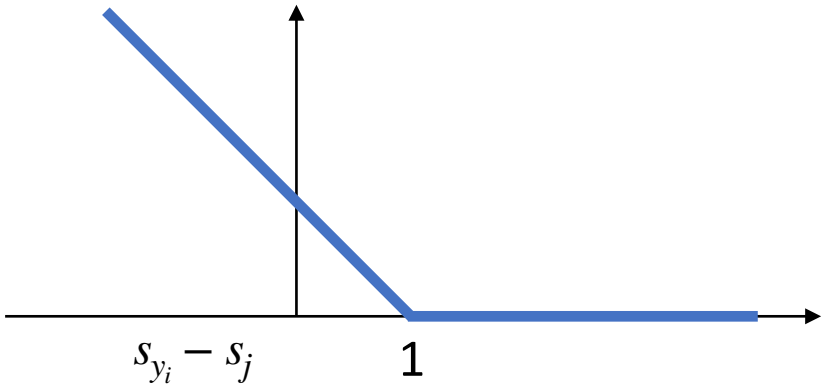
$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$

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

Loss function for SVM

- Loss over a single example ($s = f(x_i, W)$):

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



Example code:

```
def L_i(W,x,y):
    scores = W.dot(x) # score vector
    margins = np.maximum(0, scores - scores[y] + 1) # all margins
    hinge_loss_i = np.sum(margin) - margin[y] # exclude itself
    return hinge_loss_i
```

- Loss over entire datasets (averaged):

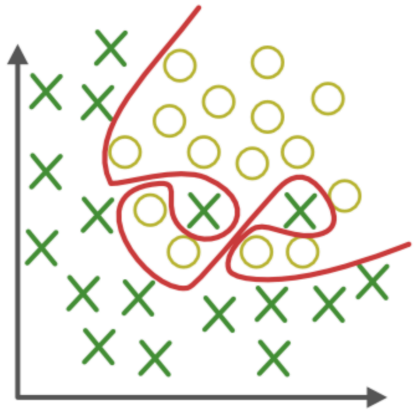
$$L(W) = \frac{1}{N} \sum_{i=1}^{N} L_i(f(x_i, W), y_i)$$

Loss function for SVM

- Minimization training loss

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

  - May overfit training data
  - Perform badly **on test data**



Over-fitting

- Add regularizations to make model simpler

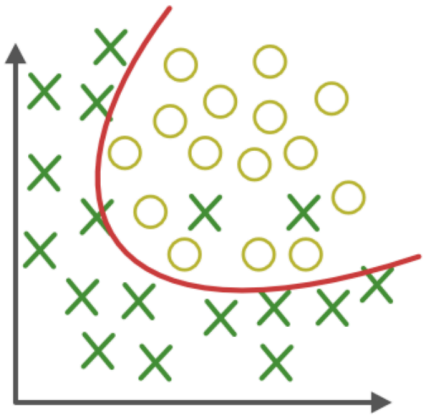
$$L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i) + \lambda R(W)$$

Weight decay $R(W) = \sum_i \sum_j W_{i,j}^2$

L_1 normalization

Dropout

Batch normalization

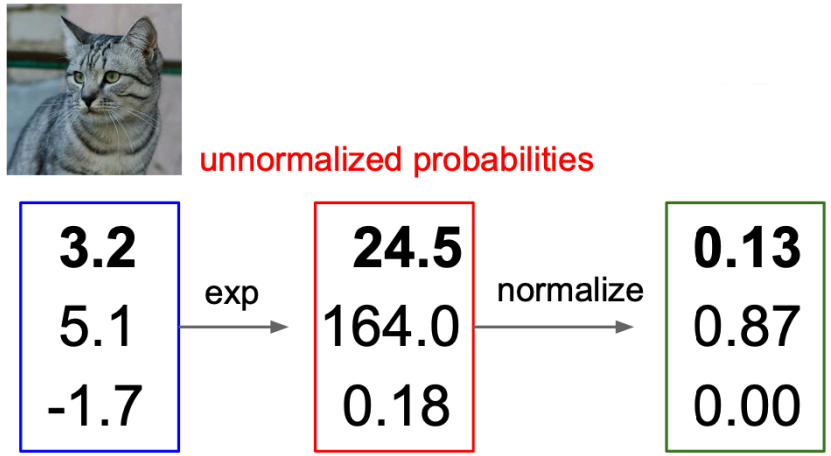


Appropirate-fitting

Multinomial Logistic Regression

- Softmax classifier
  - Score: $s = f(x_i, W) = Wx_i$
  - Softmax probability (normalized):

$$P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$$



- Negative log likelihood loss (cross-entropy loss)

$$L_i = -\log P(Y = y_i | X = x_i) = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

Increase likelihood over correct class!

- Total loss function (in sum)

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

Optimization in ML

- Linear classification, SVM, logistic regression, etc.
  - A averaged loss measures the classifier quality
- Find **lowest** loss —> find best classifier

Random search lower loss?

```
for num in xrange(1000):
  W = np.random.randn(10, 3073) * 0.0001 # generate random parameters
  loss = L(X_train, Y_train, W) # get the loss over the entire training set
  if loss < bestloss: # keep track of the best solution
    bestloss = loss
    bestW = W
```

0.15 accuracy << 0.95 SOTA

Picture and example from stanford cs231n

Optimization in ML

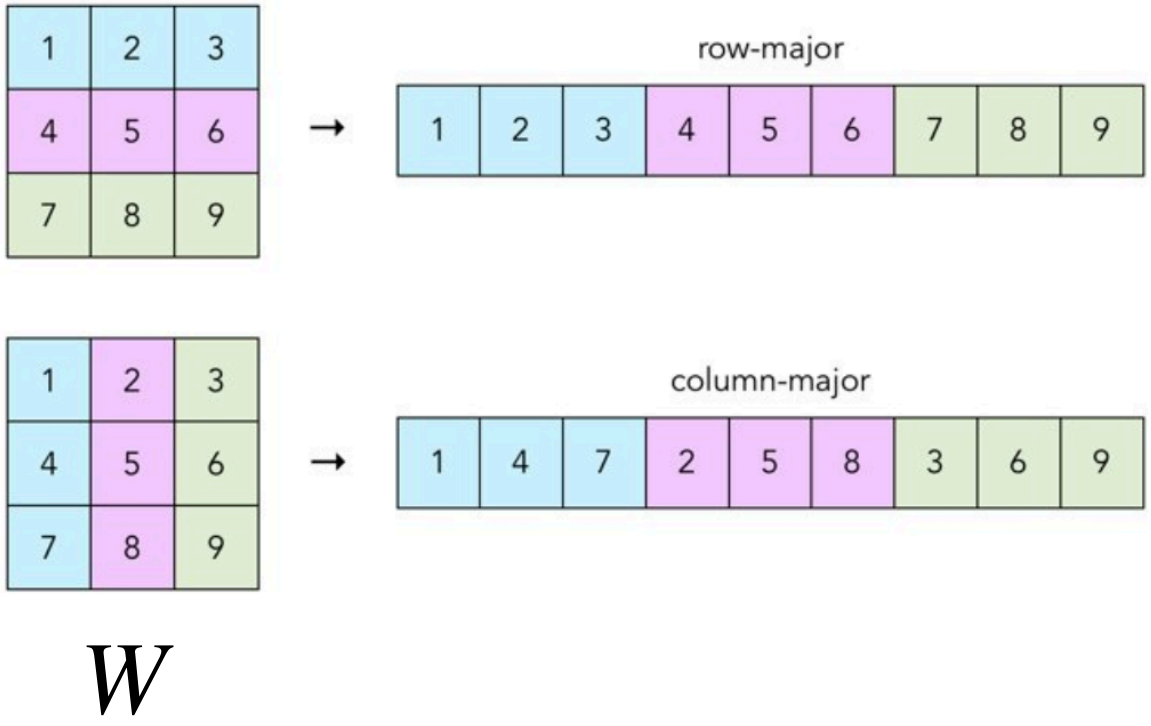
- Use slope information

- Gradient: direction to decrease loss!
  - $\nabla L(W) = \frac{dL(W)}{dW}$: compute **derivative** of $L(W)$ w.r.t. W
  - Deep learning with NNs: use automatic differentiation (e.g., backpropagation)

```
from torch.autograd import grad
grads = torch.autograd.grad(output, inputs, grad_outputs=grad_outputs, allow_unused=True,
                            retain_graph=retain_graph, create_graph=create_graph)
```

Gradient Methods in ML

- Gradient decent (GD)
  - Let $w \in \mathbb{R}^d$ be model parameters to be optimized



Gradient Methods in ML

- Gradient decent (GD)
  - Let $w \in \mathbb{R}^d$ be model parameters to be optimized

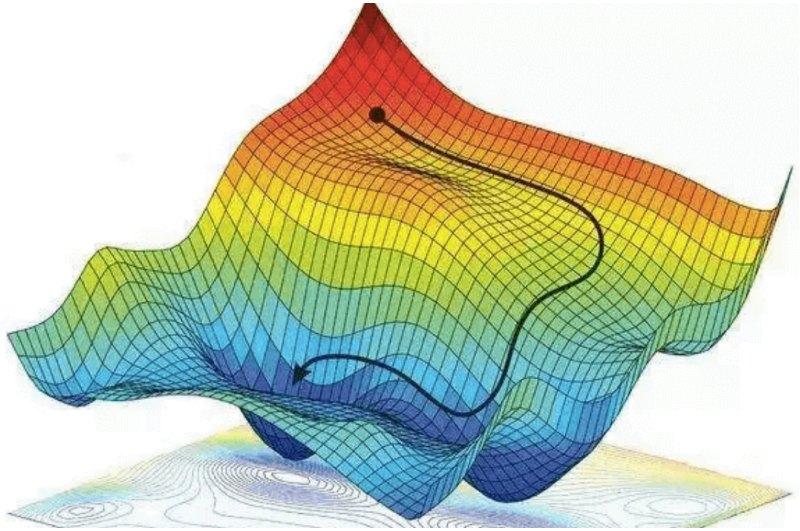
Choose initialization point $w_0 \in \mathbb{R}^d$. Repeat

$$w_{k+1} = w_k - s_k \nabla L(w_k)$$

Stop if some accuracy is achieved!

- s_k : stepsize at k^{th} iteration

How it works?



From easyai.tech

Gradient Methods in ML

- Drawbacks of GD in Modern ML
  - Recall our loss: $$L(w) = \frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, w), y_i) + \lambda R(w)$$
  - GD computes gradient $\nabla L(w)$ over **all data examples** (x_i, y_i)!
  - In ML, data size is large:

MNIST (60k), CIFAR-100 (180k) and WikiText-2 (100 million) ….

Not efficient!!

Gradient Methods in ML

- **Stochastic** Gradient decent (GD)
  - Using data in batch $B_k = \{i_1, \ldots i_b\}$ to compute gradient

$$\nabla L(w; B_k) = \frac{1}{b} \sum_{i \in B_k} L_i(f(x_i, w), y_i) + \lambda R(w)$$

Choose initialization point $w_0 \in \mathbb{R}^d$. Repeat

Sample batch $B_k = \{i_1, \ldots i_b\}$

Compute stochastic gradient $\nabla L(w; B_k)$

$$w_{k+1} = w_k - s_k \nabla L(w_k; B_k)$$

Stop if some accuracy is achieved!

- Split dataset into separate batches
- One iterate uses one batch
- One epoch often **span over** entire dataset

Gradient Methods in ML

- **Stochastic** Gradient decent (SGD)
  - Example code:

```
while True:
  data_batch = sample_training_data(data, 256) # sample 256 examples
  weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
  weights += - step_size * weights_grad # perform parameter update
```

Stanford cs231n

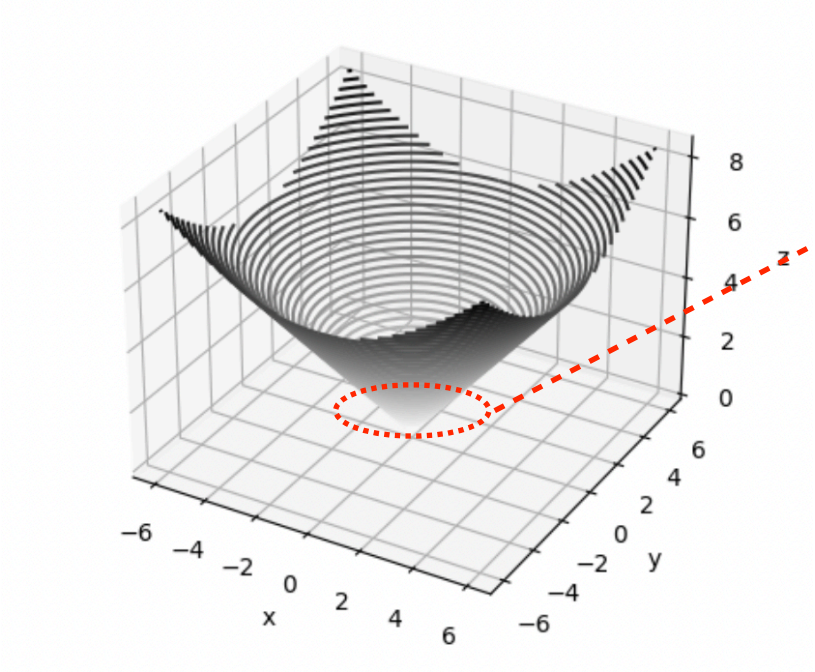
- Efficient variants (will talk later)
  - SGD with momentum
  - Adaptive stochastic methods (Adam, Adagrad, etc.)

```
torch.optim.SGD(params, lr=<required parameter>, momentum=0,
dampening=0, weight_decay=0, nesterov=False, *, maximize=False,
foreach=None) [SOURCE]
```

Accelerate via
1. History information
2. Gradient moments
3. learning rate adaptivity

Convex Functions in ML

- Linear classification, SVM, logistic regression, weight day, etc
  - Convex loss functions!!



3-D convex example from
Topics in Signal Processing

- Contain global minima (may not be unique)
- Well-shaped loss landscape

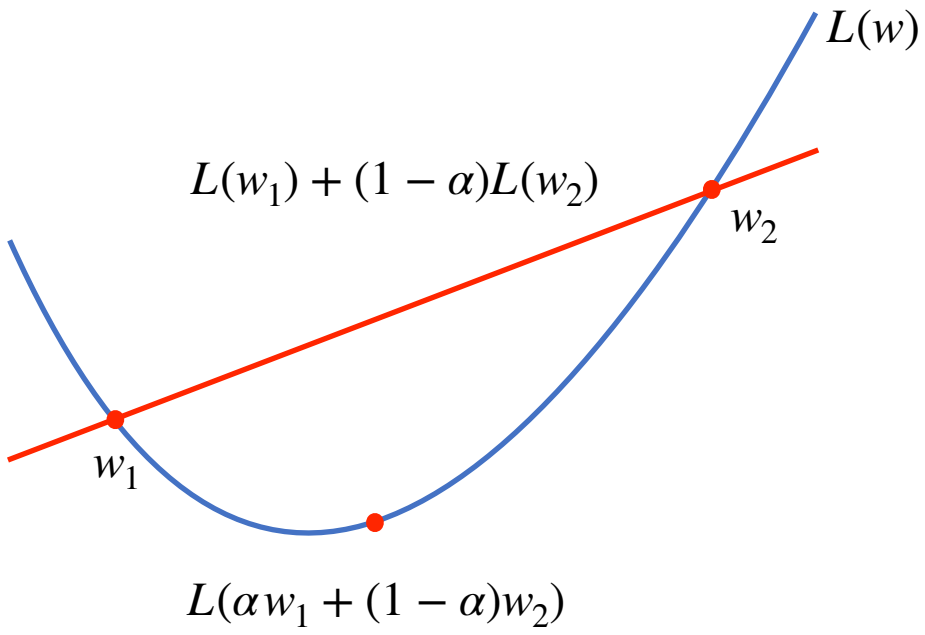
Can **gradient methods** guarantee loss decrease?

- Under such convexity.

Definition of Convex Functions

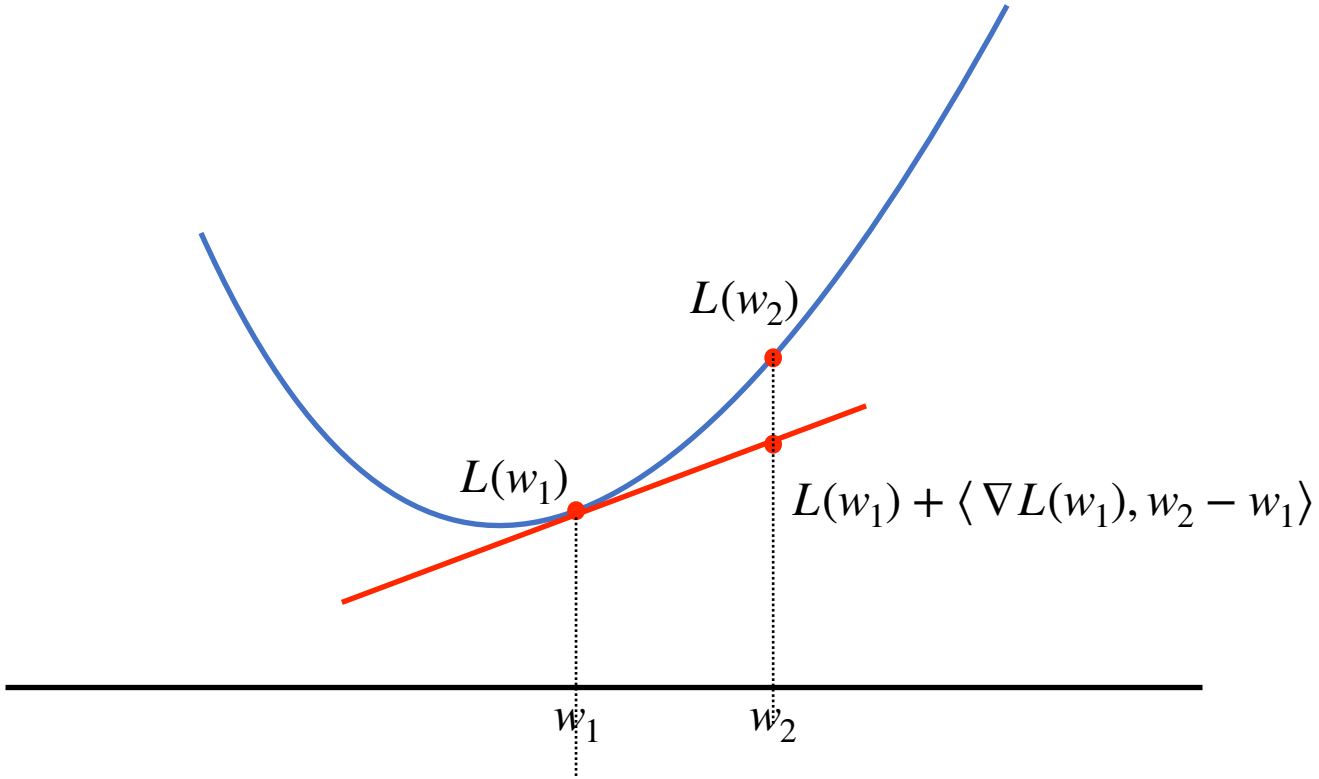
- **Definition** of Convex functions.
  - Convex function: a function $L(w) : \mathbb{R}^d \to \mathbb{R}$ is convex if for any $\alpha \in [0,1]$ and $w_1, w_2 \in \mathbb{R}^d$

$$L(\alpha w_1 + (1-\alpha)w_2) \leq \alpha L(w_1) + (1-\alpha)L(w_2)$$



Properties of Convex Functions

- Another geometrical interpretation

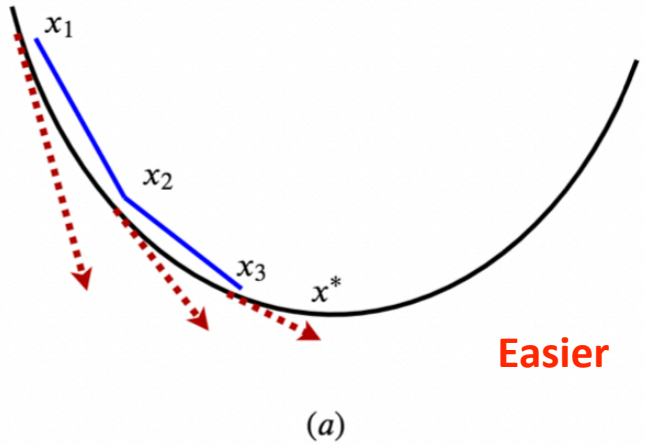


- Important relationship

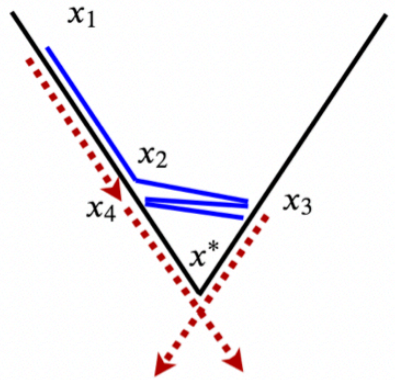
$$L(w_2) \geq L(w_1) + \langle \nabla L(w_1), w_2 - w_1 \rangle$$

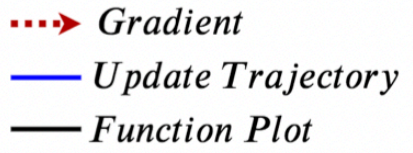
Convex Optimization in ML

- Smooth convex optimization
  - Definition of function smoothness (gradient is changed smoothly)
    - $L(w) : \mathbb{R}^d \to \mathbb{R}$ is L-**smooth** if for any $w_1, w_2 \in \mathbb{R}^d$, $\quad \|\nabla L(w_1) - \nabla L(w_2)\| \leq L\|w_1 - w_2\|.$



Easier





Picture from Ioannis's lecture note

Smooth function

Non-smooth function

Convex Optimization in ML

- Smooth convex optimization

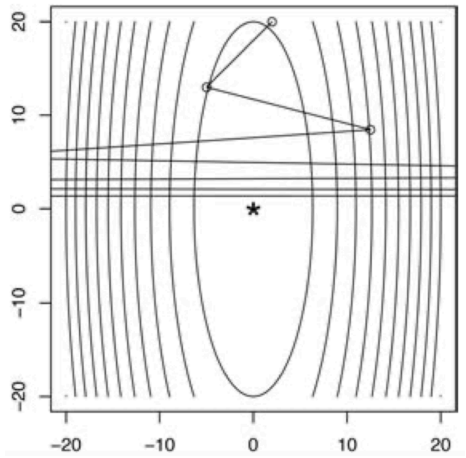
Convergence of gradient decent (GD) with **fixed stepsize** $s_k = s$

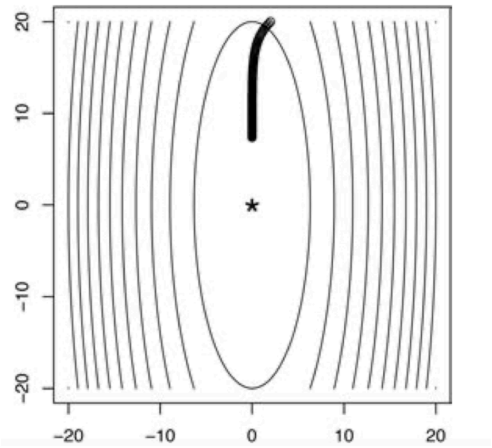
- If loss function $L(w) : \mathbb{R}^d \to \mathbb{R}$ is **convex**, **differentiable** and L-**smooth**, then GD with stepsize $s < 1/L$ satisfies

$$L(w_K) - L(w^*) \leq \frac{\|w_0 - w^*\|^2}{2sK}$$

Guaranteed solution!

Can diverge if s is too large



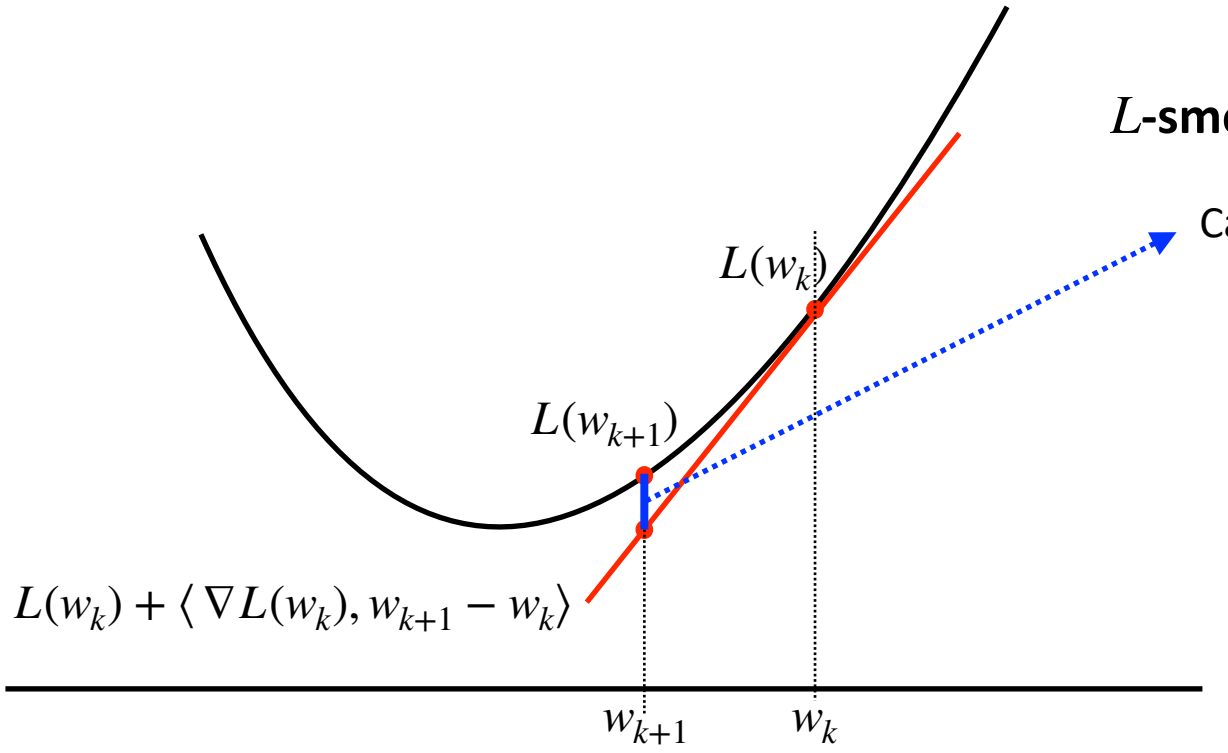


Can be slow if s is too small

How to show this theory?

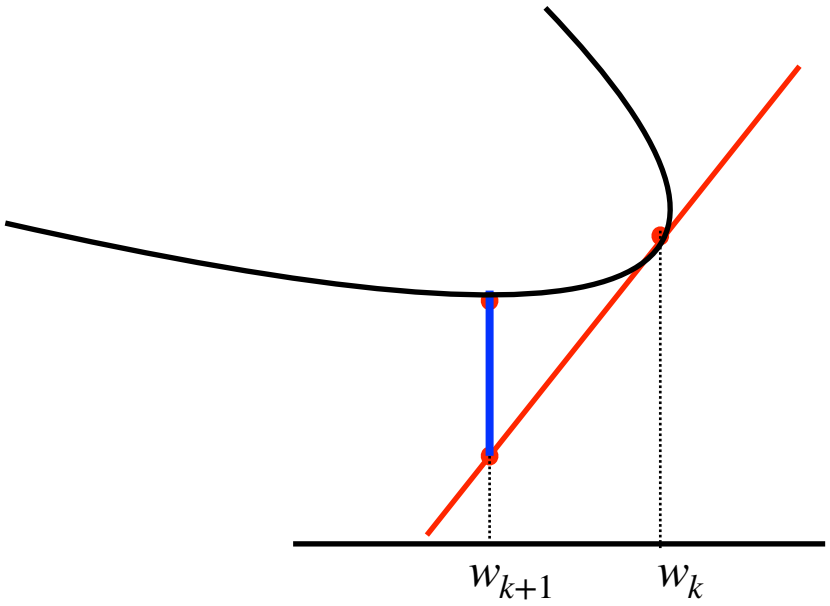
- Step 1: decent lemma by L-**smoothness**

$$L(w_{k+1}) \leq L(w_k) + \langle \nabla L(w_k), w_{k+1} - w_k \rangle + \frac{L}{2}\|w_{k+1} - w_k\|^2$$



L-**smoothness** Gradient changes slowly

Cannot be too large $\leq \frac{L}{2}\|w_{k+1} - w_k\|^2$



How to show this theory?

- Step 1: decent lemma by L-**smoothness**

$$L(w_{k+1}) \leq L(w_k) + \langle \nabla L(w_k), w_{k+1} - w_k \rangle + \frac{L}{2}\|w_{k+1} - w_k\|^2$$

GD update: $w_{k+1} = w_k - s\nabla L(w_k)$

$$L(w_{k+1}) \leq L(w_k) - s\|\nabla L(w_k)\|^2 + \frac{s^2 L}{2}\|\nabla L(w_k)\|^2$$

Stepsize choice: $s = 1/L$

$$L(w_{k+1}) \leq L(w_k) - \frac{1}{2L}\|\nabla L(w_k)\|^2$$

Guaranteed decreasing $L(w_k)$!

How to show this theory?

- Step 2: By convexity of $L(w)$

$$L(w^*) - L(w_k) \geq \langle \nabla L(w_k), w^* - w_k \rangle$$

Rearrage

$$L(w_k) - L(w^*) \leq \langle \nabla L(w_k), w_k - w^* \rangle$$

- Step 3: Combine decent lemma in **step 1** and convexity in **step 2**

Step 1 **Step 2**

$$L(w_{k+1}) - L(w^*) \leq L(w_k) - L(w^*) - \frac{1}{2L}\|\nabla L(w_k)\|^2 \leq \langle \nabla L(w_k), w_k - w^* \rangle - \frac{1}{2L}\|\nabla L(w_k)\|^2$$

$$= \frac{L}{2}(\|w_k - w^*\|^2 - \|w_{k+1} - w^*\|^2)$$

How to show this theory?

- Step 3 leads to:

$$L(w_{k+1}) - L(w^*) \leq \frac{L}{2}(\|w_k - w^*\|^2 - \|w_{k+1} - w^*\|^2)$$

- Step 4: **telescope** step 3

$$\sum_{k=0}^{K-1} (L(w_{k+1}) - L(w^*)) \leq \frac{L}{2}\|w_0 - w^*\|^2$$

- Final step: using decreasing $L(w_k)$ (established in step 1)

$$L(w_K) - L(w^*) \leq \frac{1}{K}\sum_{k=0}^{K-1} (L(w_{k+1}) - L(w^*)) \leq \frac{L\|w_0 - w^*\|^2}{2K}$$

Happy ending!

More Results in Convex Optimization

- GD convergence in convex functions:

$$L(w_K) - L(w^*) \leq \frac{\|w_0 - w^*\|^2}{2sK}$$

- Can this $\frac{1}{K}$ rate be **improved?**

Yes! Optimal rate is $\frac{1}{K^2}$ (Theorem 2.1.6 in Nesterov's lecture notes)

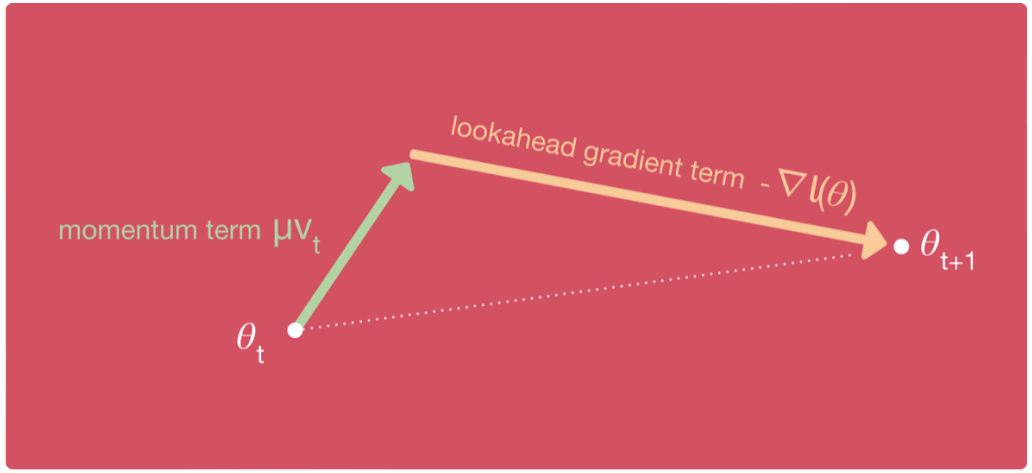
- Can we develop **optimal** and **faster** algorithm?

Yes! Nesterov's momentum!! (widely used in deep learning as well)

More Results in Convex Optimization

- Nesterov's momentum (theory motived!!)

Sutskever, Martens et al. "On the importance of initialization and momentum in deep learning," 2013

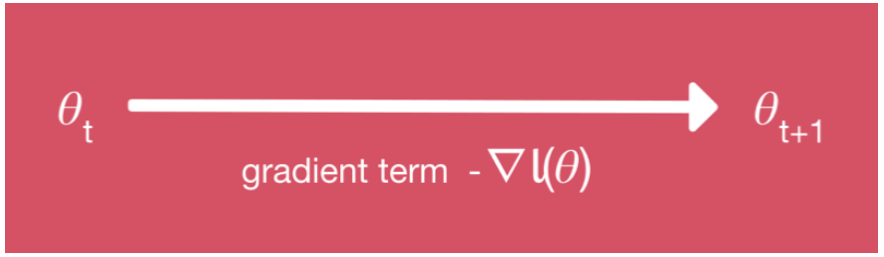


$$v_{t+1} = \mu v_t - \eta \nabla l(\theta + \mu v_t)$$
$$\theta_{t+1} = \theta_t + v_{t+1}$$

Convergence analysis: Section 2.2 in Nesterov's lecture notes

- Auxiliary variable & function analysis

- Compared to gradient decent (GD)

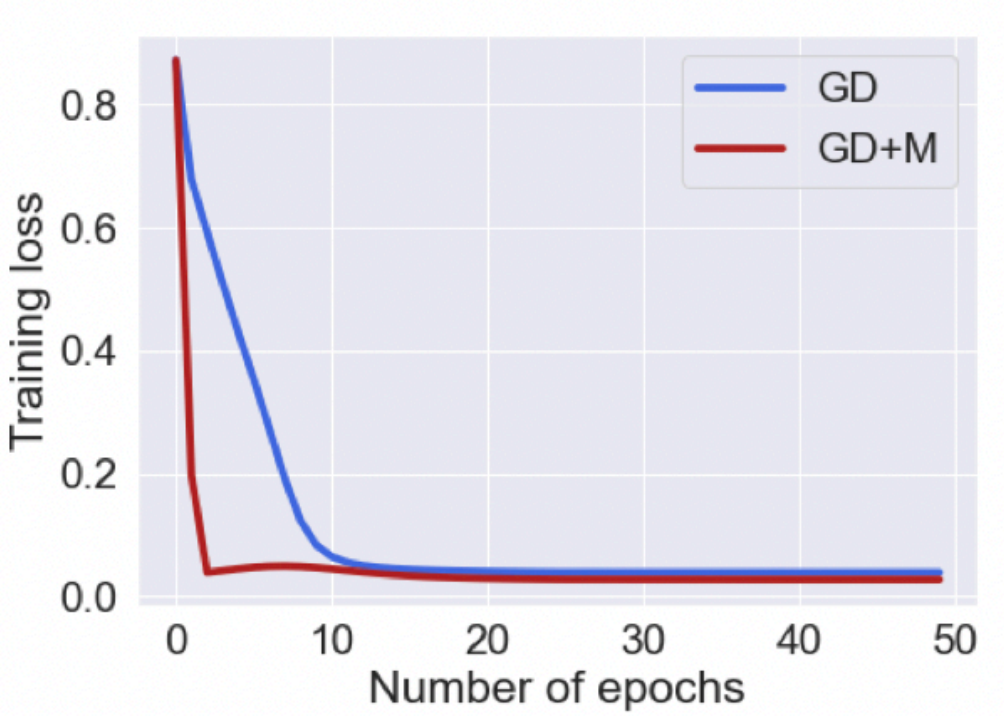


$$\theta_{t+1} = \theta_t - \eta \nabla l(\theta)$$

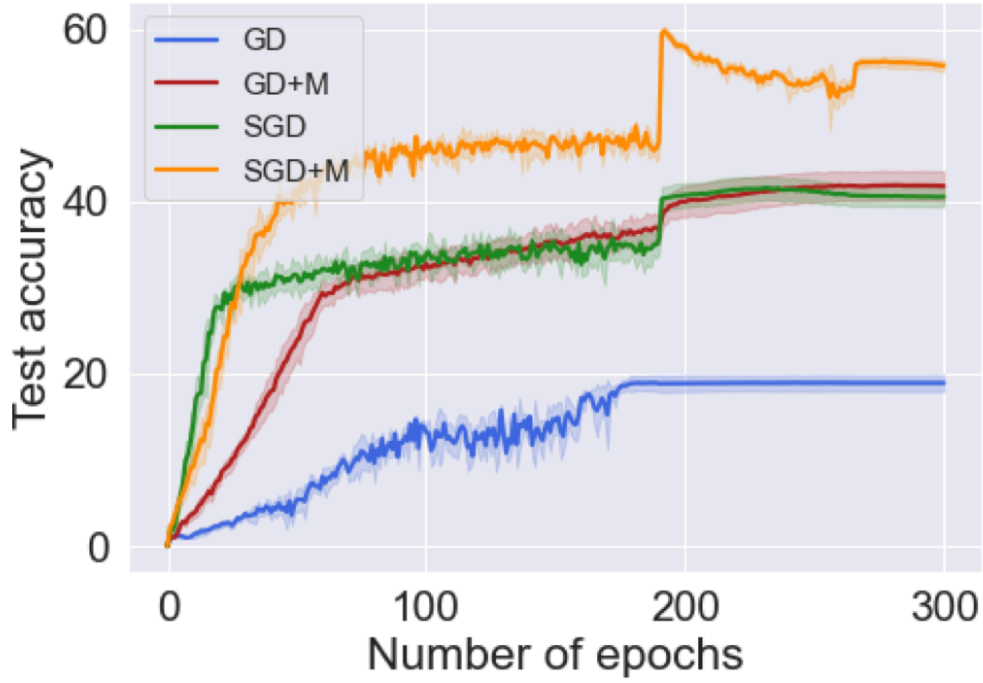
More Results in Convex Optimization

- Comparison: (S)GD and Nesterov's momentum

CIFAR 10 with VGG-19



CIFAR 100 with VGG-19



Jelassi & Li. *Towards understanding how momentum improves generalization in deep learning.* ICML 2022.

Summary

- Classic machine learning problems
  - Linear classification, SVM, logistic regression
  - Convex loss functions over data examples
- Convex optimization in machine learning
  - Gradient methods: (stochastic) gradient descent
  - Properties of convex functions
  - Convergence analysis of gradient descent
- More results in optimization
  - Nesterov momentum in training neural networks