Machine Learning

Regression & Gradient Descent

Dr. Shuang LIANG

Recall: Terminology

- Data
  - Data set, feature, dimentionality, label, sample…
- Train & Test
- Task
  - By prediction target
  - By label

Recall: Error and Overfitting

- Error rate/Accuracy
  - $E = \frac{the\ number\ of\ misclassified\ samples\ (a)}{the\ number\ of\ all\ samples\ (m)} = \frac{a}{m}$
- Error
  - Train/Test/Generalization error
- Overfitting
  - Small loss on training data, large loss on testing data
  - Can’t be avoided completely

Recall: Evaluation Methods

- Hold-out
- Cross Validation
- Bootstrapping

Recall: Performance Measure

MSE for Regression $$E(f;D)=\frac{1}{m}\sum_{i=1}^{m}\left(f\left(\boldsymbol{x}_i\right)-y_i\right)^2$$

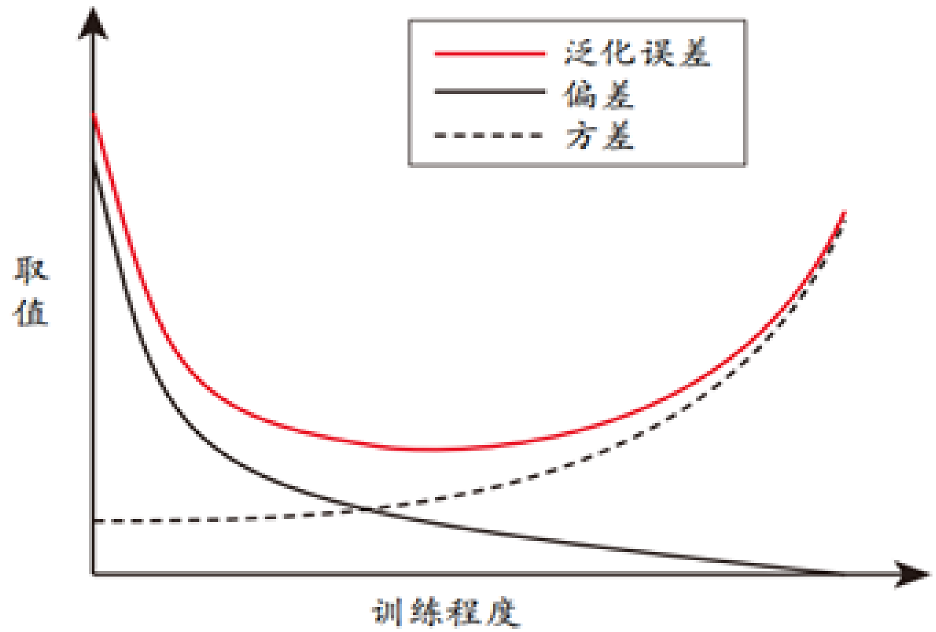
Accuracy for Classification
$$\begin{aligned}\operatorname{acc}(f;D) &= \frac{1}{m}\sum_{i=1}^{m}\mathbb{I}\left(f\left(\boldsymbol{x}_i\right)=y_i\right)\\ &= 1-E(f;D)\ .\end{aligned}$$

Precision $P=\dfrac{TP}{TP+FP}$

Recall $R=\dfrac{TP}{TP+FN}$

$$F1=\frac{2*P*R}{P+R}=\frac{2*TP}{the\ number\ of\ samples+TP-TN}$$

Recall: Bias and variance



泛化误差与偏差、方差的关系示意图

Large bias	Large variance
Add more features as input	More data
A more complex model	Regularization

Today's Topics

- Regression
- Linear Regression
- Gradient Descent
- Advanced Regression Methods

Today’s Topics

- ***Regression***
- Linear Regression
- Gradient Descent
- Advanced Regression Methods

Different types of functions

Regression: The function outputs a scalar.

Predict PM2.5

PM2.5 today → f → PM2.5 of tomorrow
temperature → f
Concentration of O_3 → f

Classification: Given options (**classes**), the function outputs the correct one.

Spam filtering

→ f → Yes/No

Structured Learning

create something with structure (image, document)

Regression, Classification

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"boy is doing backflip on wakeboard."

"girl in pink dress is jumping in air."

"black and white dog jumps over bar."

"young girl in pink shirt is swinging on swing."

"man in blue wetsuit is surfing on wave."

Regression

- Stock Market Forecast

$f($  $) =$ Dow Jones Industrial Average at tomorrow

- Self-driving Car

$f($  $) =$ 方向盘角度

- Recommendation

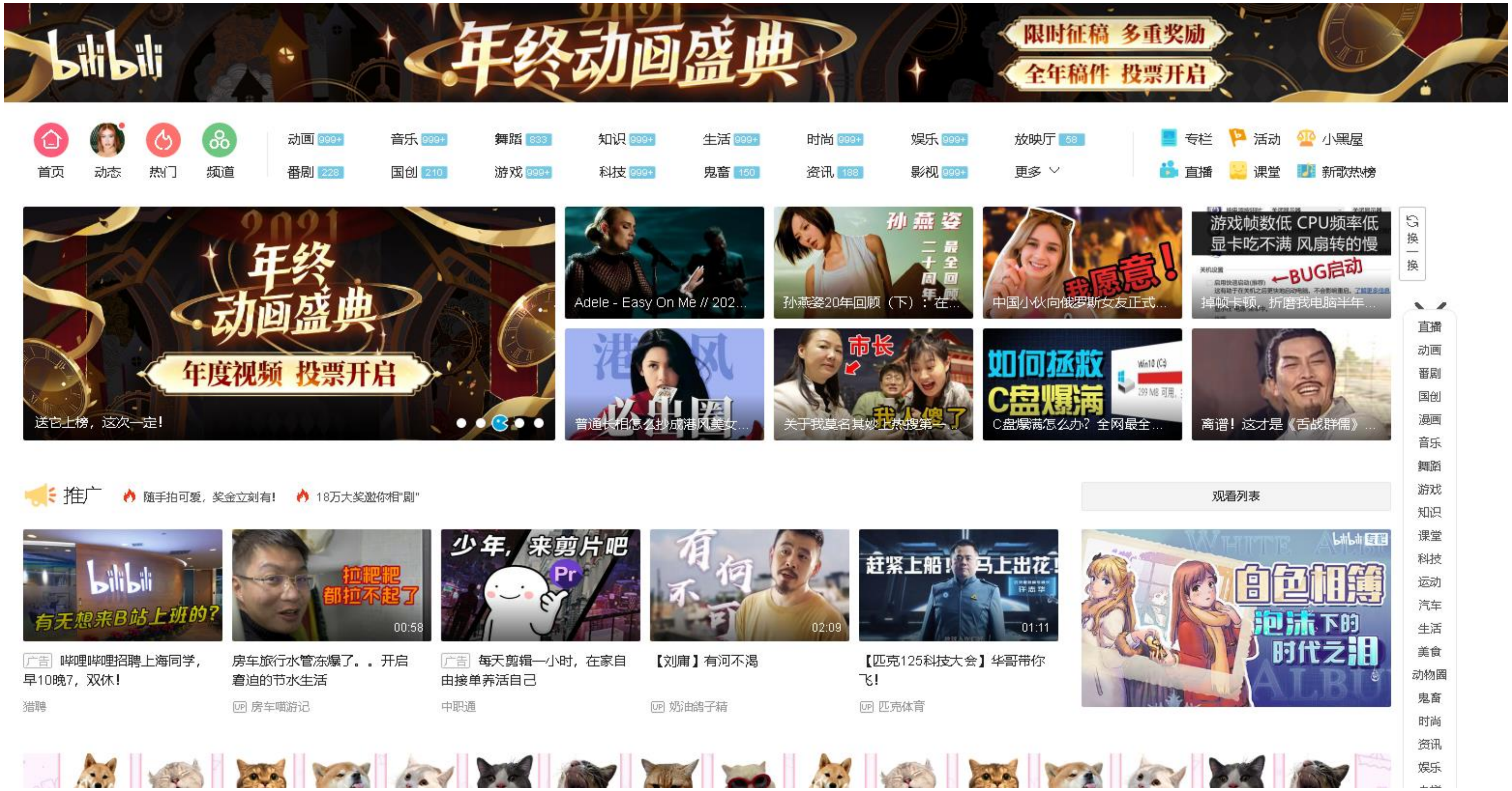
$f($ User A Commodity B $) =$ 购买可能性

Today's Topics

- Regression
- ***Linear Regression***
- Gradient Descent
- Advanced Regression Methods

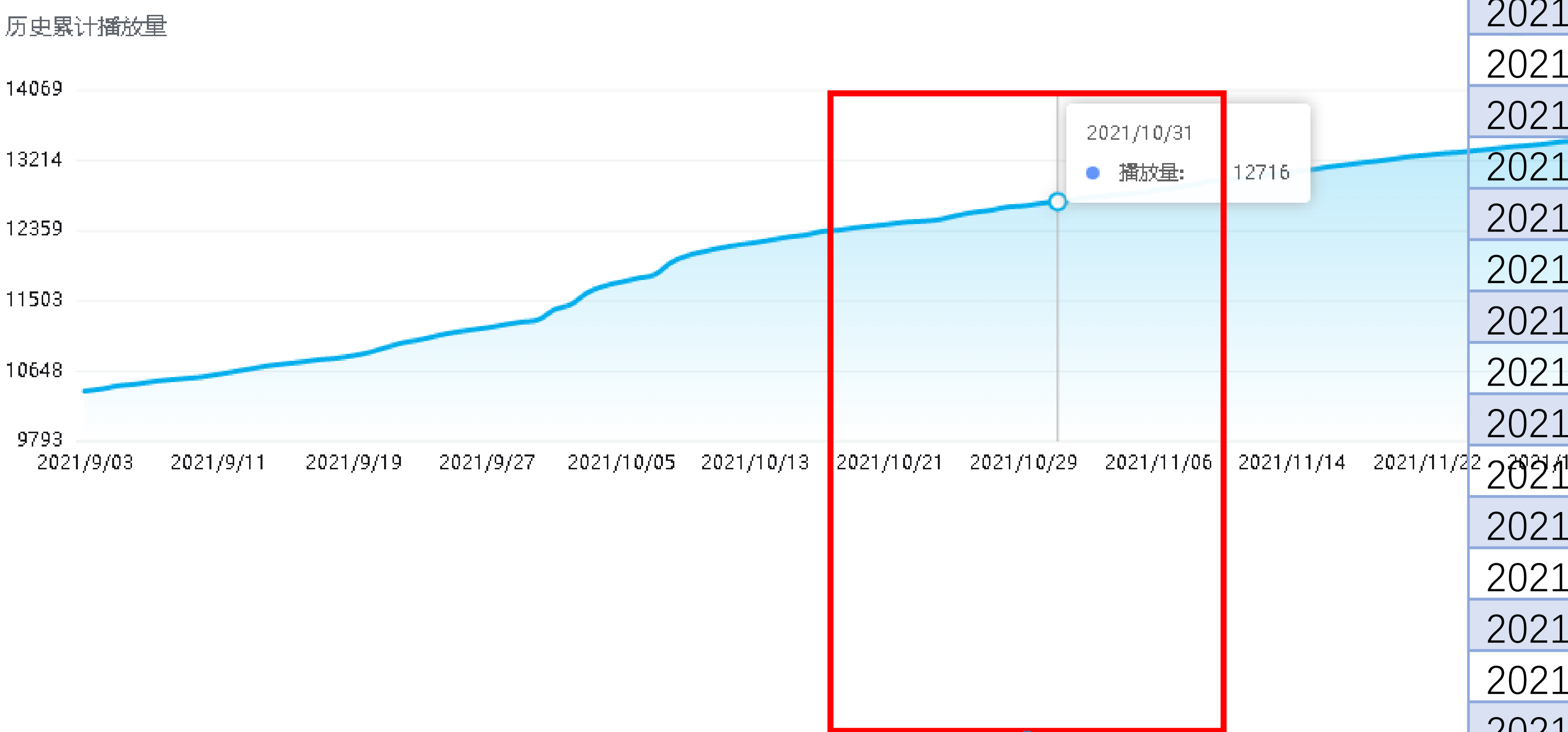
How to find a good function?

- **A bilibili case study**



How to find a good function?

- **A bilibili case study**



时间	播放量
2021-10-21	12442
2021-10-22	12465
2021-10-23	12478
2021-10-24	12499
2021-10-25	12547
2021-10-26	12584
2021-10-27	12610
2021-10-28	12648
2021-10-29	12664
2021-10-30	12692
2021-10-31	12716
2021-11-01	12740
2021-11-02	12769
2021-11-03	12790
2021-11-04	12808
2021-11-05	12825
2021-11-06	12863

The function we want to find …

$$y = f\Big($$

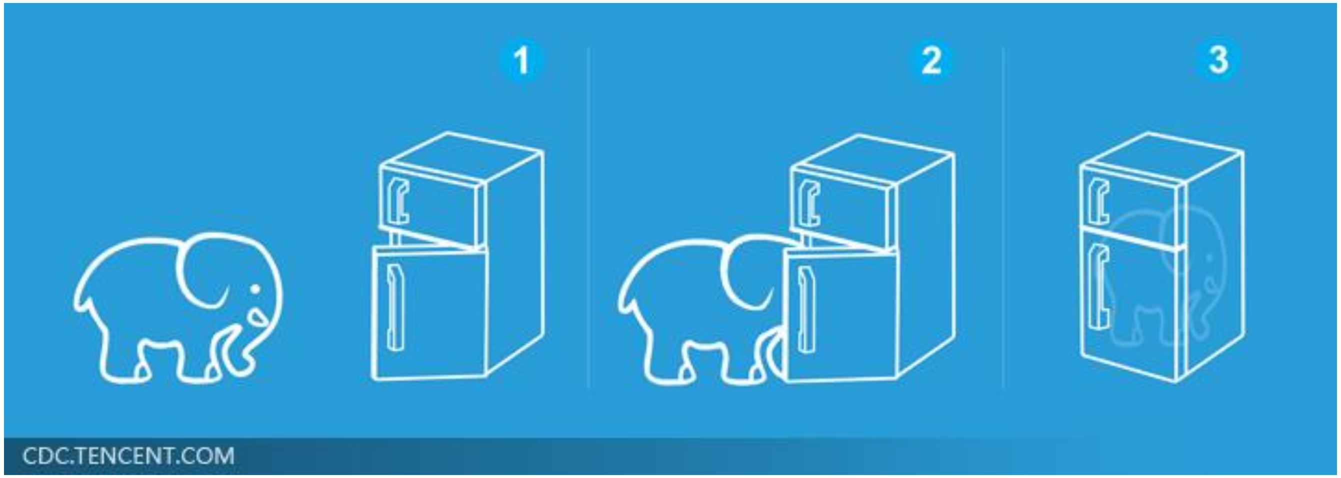
时间	播放量
2021-10-21	12442
2021-10-22	12465
2021-10-23	12478
2021-10-24	12499
2021-10-25	12547
2021-10-26	12584
2021-10-27	12610
2021-10-28	12648
2021-10-29	12664
2021-10-30	12692
2021-10-31	12716
2021-11-01	12740
2021-11-02	12769
2021-11-03	12790
2021-11-04	12808
2021-11-05	12825
2021-11-06	12863

$$\Big)$$

no. of views
on 11/18

Typical process of ML

Step 1: function with unknown param → Step 2: define loss from training data → Step 3: optimization



Step1: Function with Unknown Parameters

$$y = f(\quad)$$

时间	播放量
2021-10-21	12442
2021-10-22	12465
2021-10-23	12478
2021-10-24	12499
2021-10-25	12547
2021-10-26	12584
2021-10-27	12610
2021-10-28	12648
2021-10-29	12664
2021-10-30	12692
2021-10-31	12716
2021-11-01	12740
2021-11-02	12769
2021-11-03	12790
2021-11-04	12808
2021-11-05	12825
2021-11-06	12863

Model $y = b + wx_1$ based on domain knowledge

feature

y: no. of views on 11/18, x_1: no. of views on 11/17

w and b are unknown parameters (learned from data)

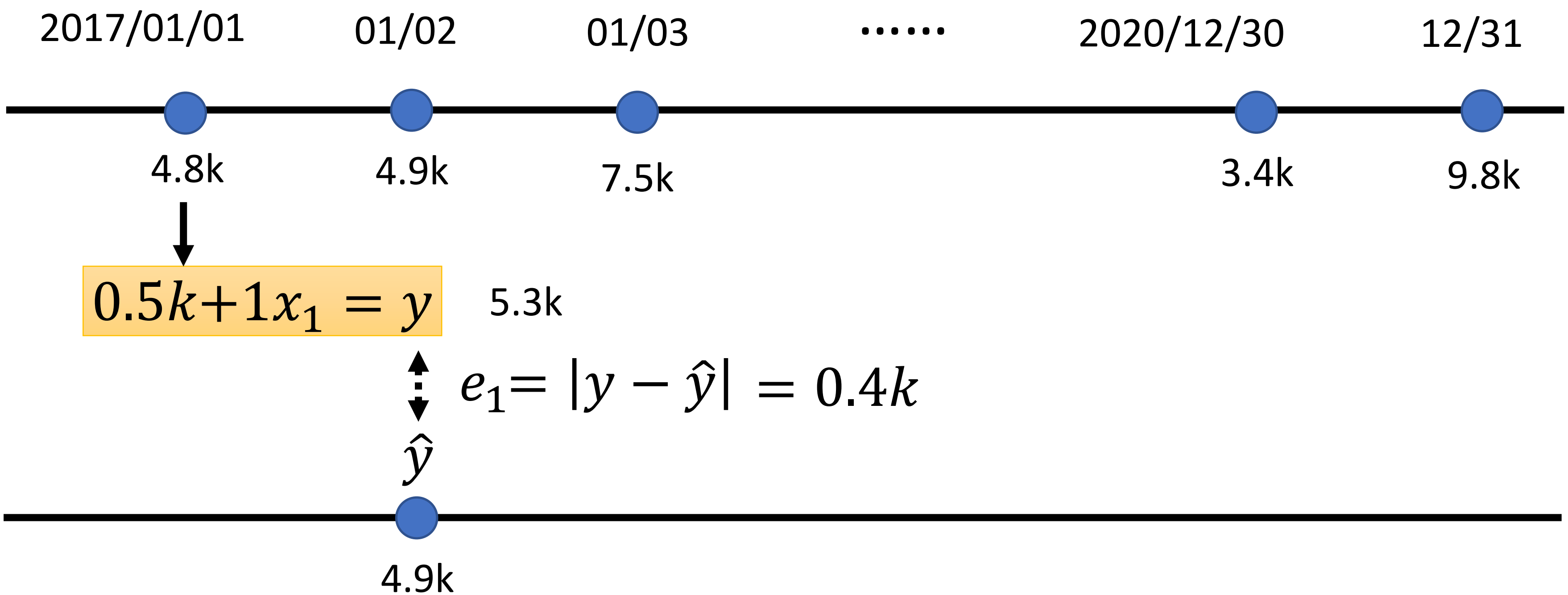
weight **bias**

Step2: Define Loss from Training Data

- Loss is a function of parameters $L(b, w)$
- Loss: how good a set of values is.

$L(0.5k, 1)$ $\quad y = b + wx_1 \longrightarrow y = 0.5k + 1x_1$ How good it is?

Data from 2017/01/01 – 2020/12/31

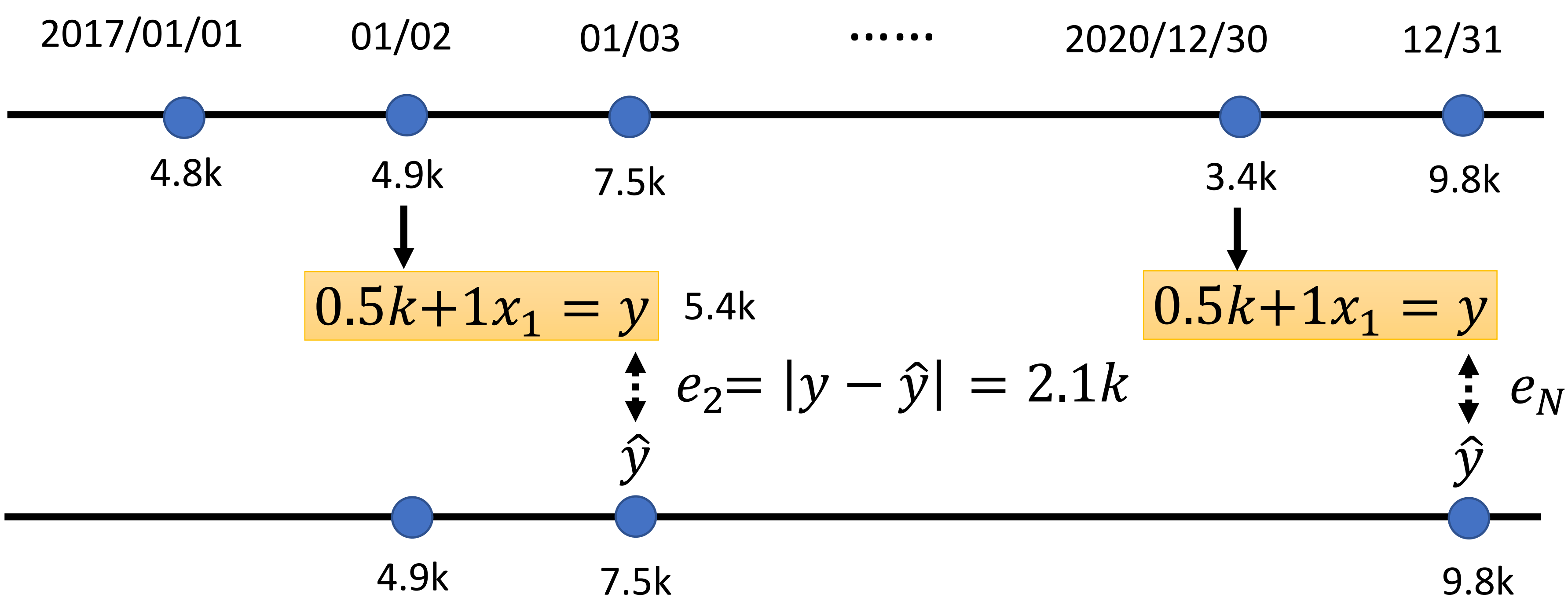


Step2: Define Loss from Training Data

- Loss is a function of parameters $L(b, w)$
- Loss: how good a set of values is.

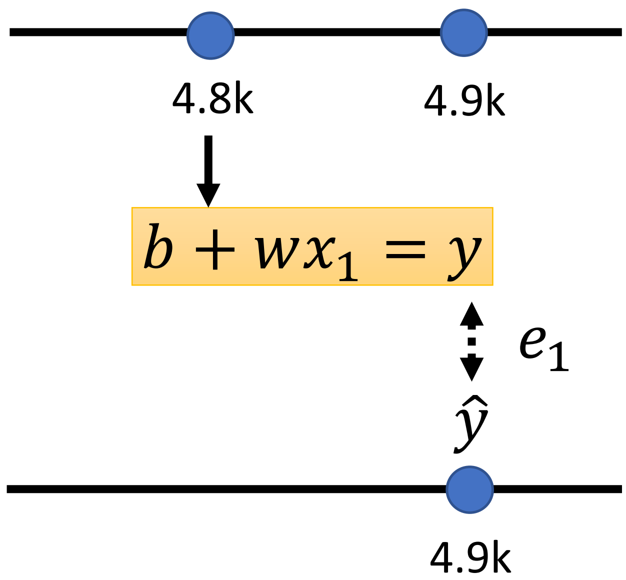
$L(0.5k, 1)$ $\quad y = b + wx_1 \longrightarrow y = 0.5k + 1x_1$ How good it is?

Data from 2017/01/01 – 2020/12/31



Step2: Define Loss from Training Data

- Loss is a function of parameters $L(b, w)$
- Loss: how good a set of values is.



Loss: $L = \frac{1}{N}\sum_{n} e_n$

$e = |y - \hat{y}|$ L is mean absolute error (**MAE**)

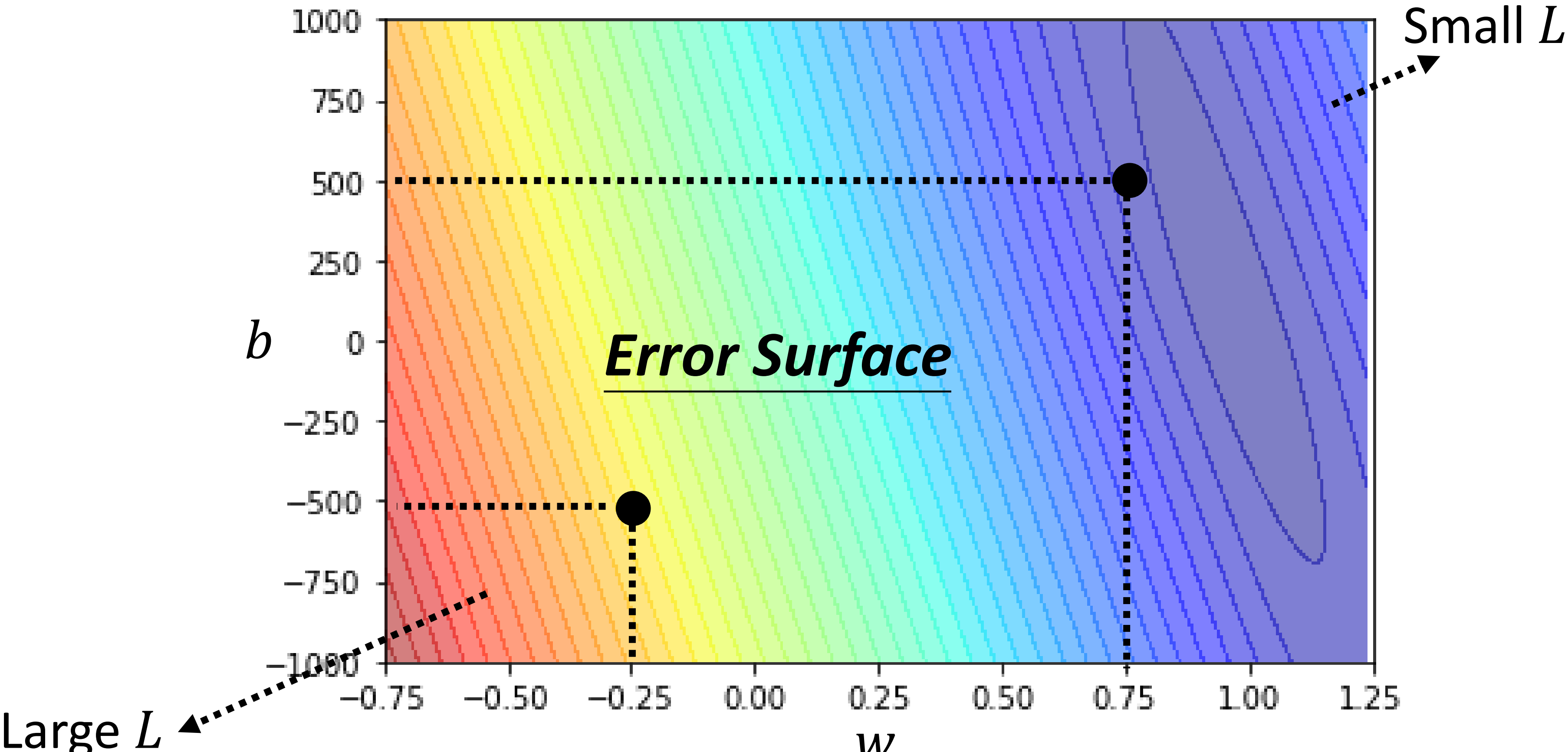
$e = (y - \hat{y})^2$ L is mean square error (**MSE**)

If y and $\hat{y}$ are both probability distributions ➡ Cross-entropy

Step2: Define Loss from Training Data

- Loss is a function of parameters $L(b, w)$
- Loss: how good a set of values is.

Model $y = b + wx_1$



Step3: Optimization

Gradient Descent
梯度下降

In 1-dimension, the derivative of a function:

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

In multiple dimensions, the gradient is the vector of (partial derivatives) along each dimension.
The slope in any direction is the dot product of the direction with the gradient.
The direction of steepest descent is the negative gradient.

Practice

Try to calculate the gradient!

$$f(x_1, x_2, x_3) = \ln(1 + \exp(-2x_1 + 3x_2 - 4x_3))$$

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \frac{\partial f}{\partial x_3} \end{bmatrix} = \begin{bmatrix} -\frac{2e^{-2x_1+3x_2-4x_3}}{1+e^{-2x_1+3x_2-4x_3}} \\ \frac{3e^{-2x_1+3x_2-4x_3}}{1+e^{-2x_1+3x_2-4x_3}} \\ -\frac{4e^{-2x_1+3x_2-4x_3}}{1+e^{-2x_1+3x_2-4x_3}} \end{bmatrix}$$

Today's Topics

- Regression
- Linear Regression
- ***Gradient Descent***
- Advanced Regression Methods

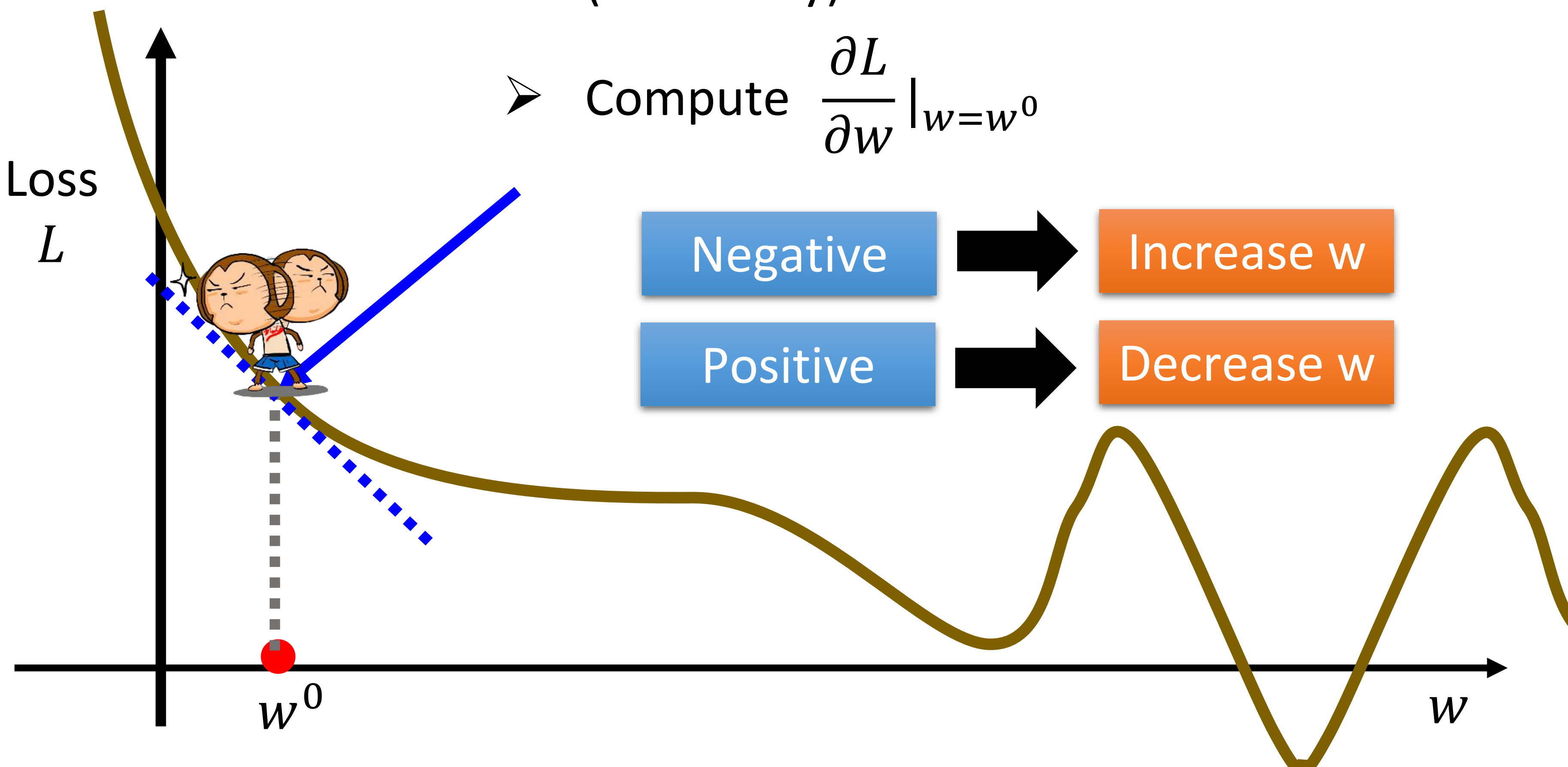
Gradient Descent

w^{*} [redacted] $= arg \min_{w}$ [redacted] L

- (Randomly) Pick an initial value w^0
- Compute $\frac{\partial L}{\partial w}|_{w=w^0}$

Negative → Increase w

Positive → Decrease w



Gradient Descent

w^* [redacted] $= arg \min_{w}$ [redacted] L

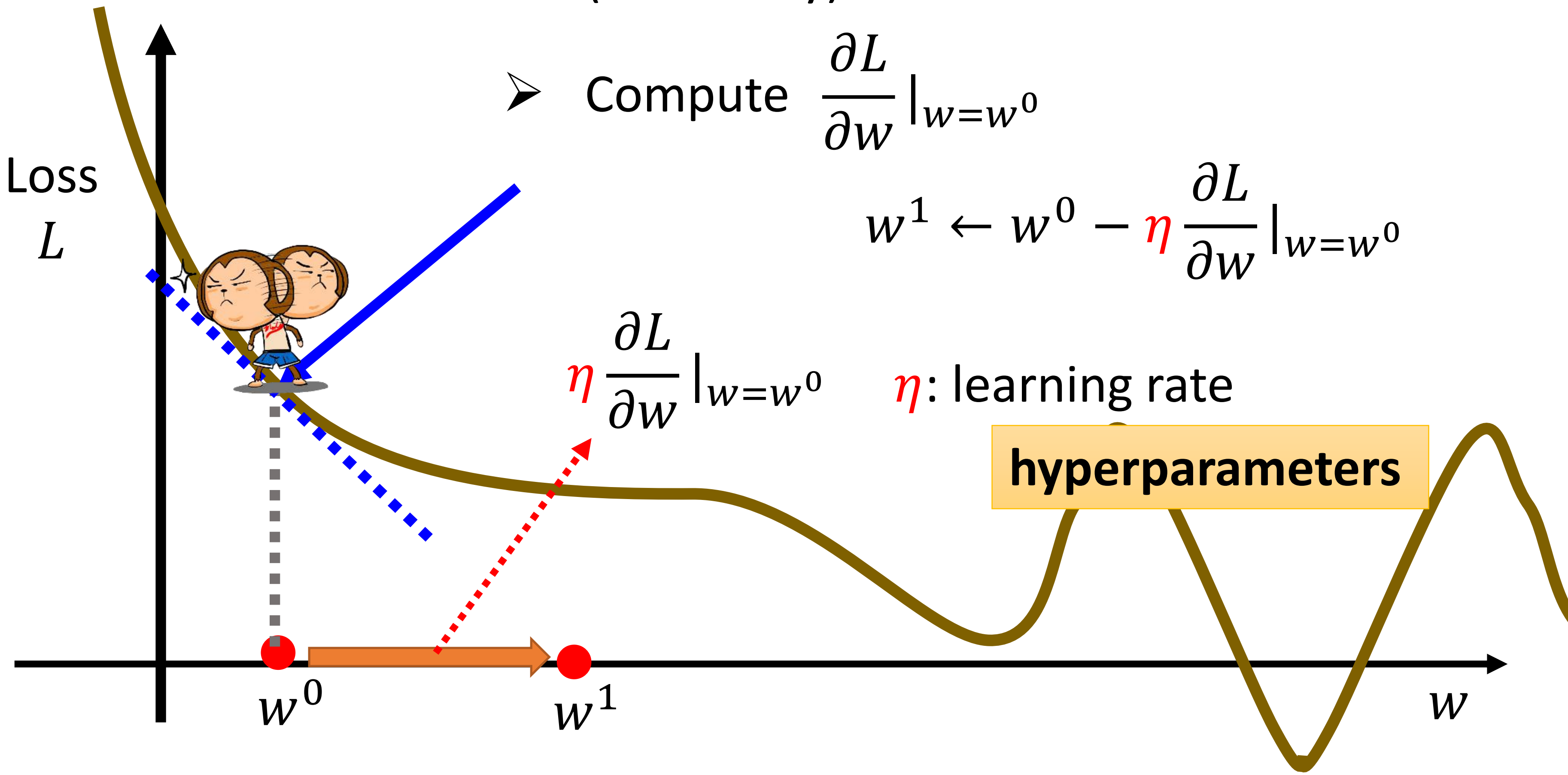
➢ (Randomly) Pick an initial value w^0

➢ Compute $\frac{\partial L}{\partial w}|_{w=w^0}$

$$w^1 \leftarrow w^0 - \eta \frac{\partial L}{\partial w}|_{w=w^0}$$

η: learning rate

hyperparameters



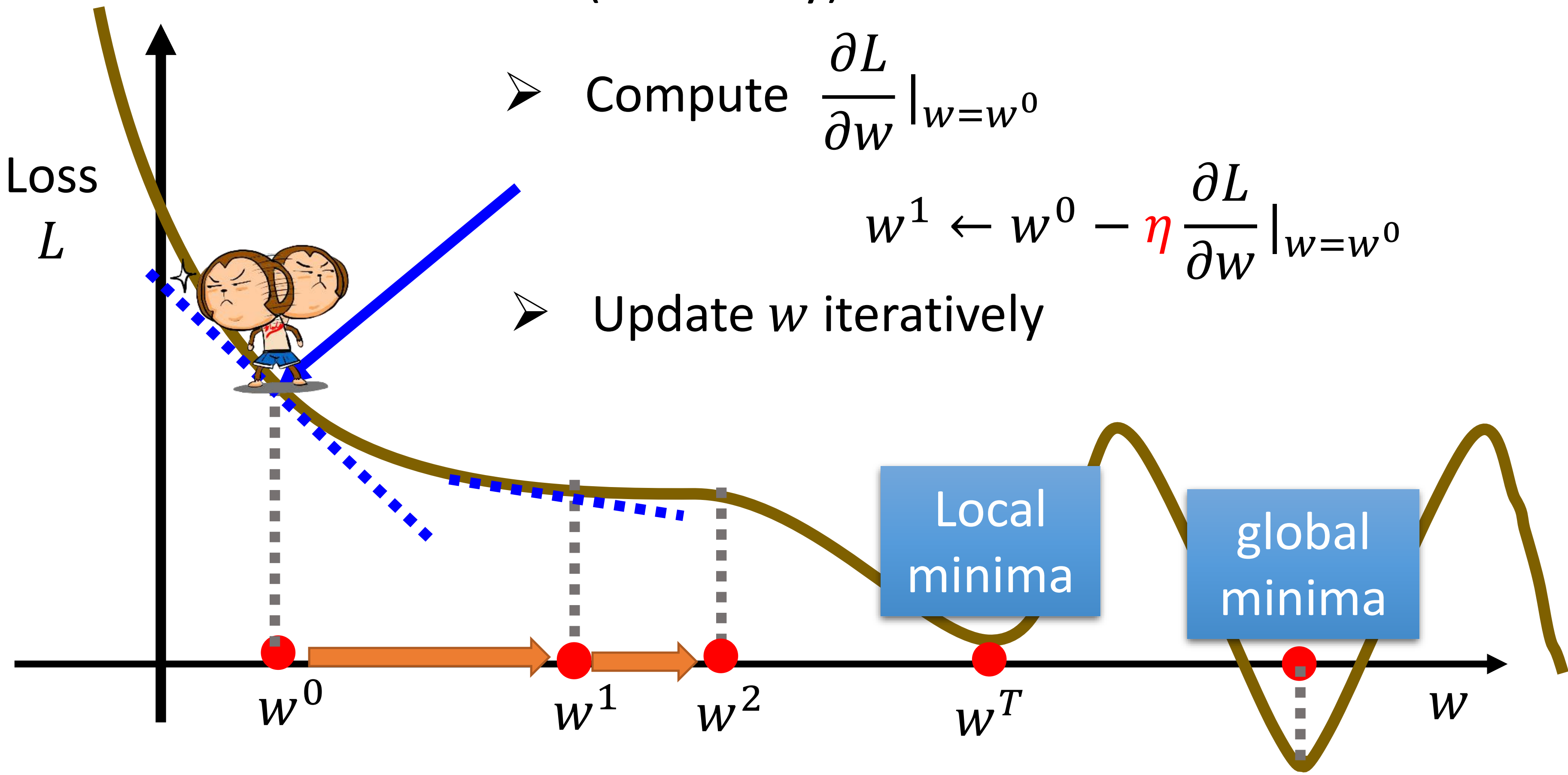
Gradient Descent

w^* [redacted] $= arg \min_{w}$ [redacted] L

- (Randomly) Pick an initial value w^0
- Compute $\frac{\partial L}{\partial w}|_{w=w^0}$

$$w^1 \leftarrow w^0 - \eta \frac{\partial L}{\partial w}|_{w=w^0}$$

- Update w iteratively



Gradient Descent

$$w^*, b^* = arg \min_{w,b} L$$

- (Randomly) Pick initial values w^0, b^0
- Compute

$$\boxed{\begin{gathered}\frac{\partial L}{\partial w}|_{w=w^0,b=b^0} \\ \frac{\partial L}{\partial b}|_{w=w^0,b=b^0}\end{gathered}}$$

$$w^1 \leftarrow w^0 - \eta \frac{\partial L}{\partial w}|_{w=w^0,b=b^0}$$

$$b^1 \leftarrow b^0 - \eta \frac{\partial L}{\partial b}|_{w=w^0,b=b^0}$$

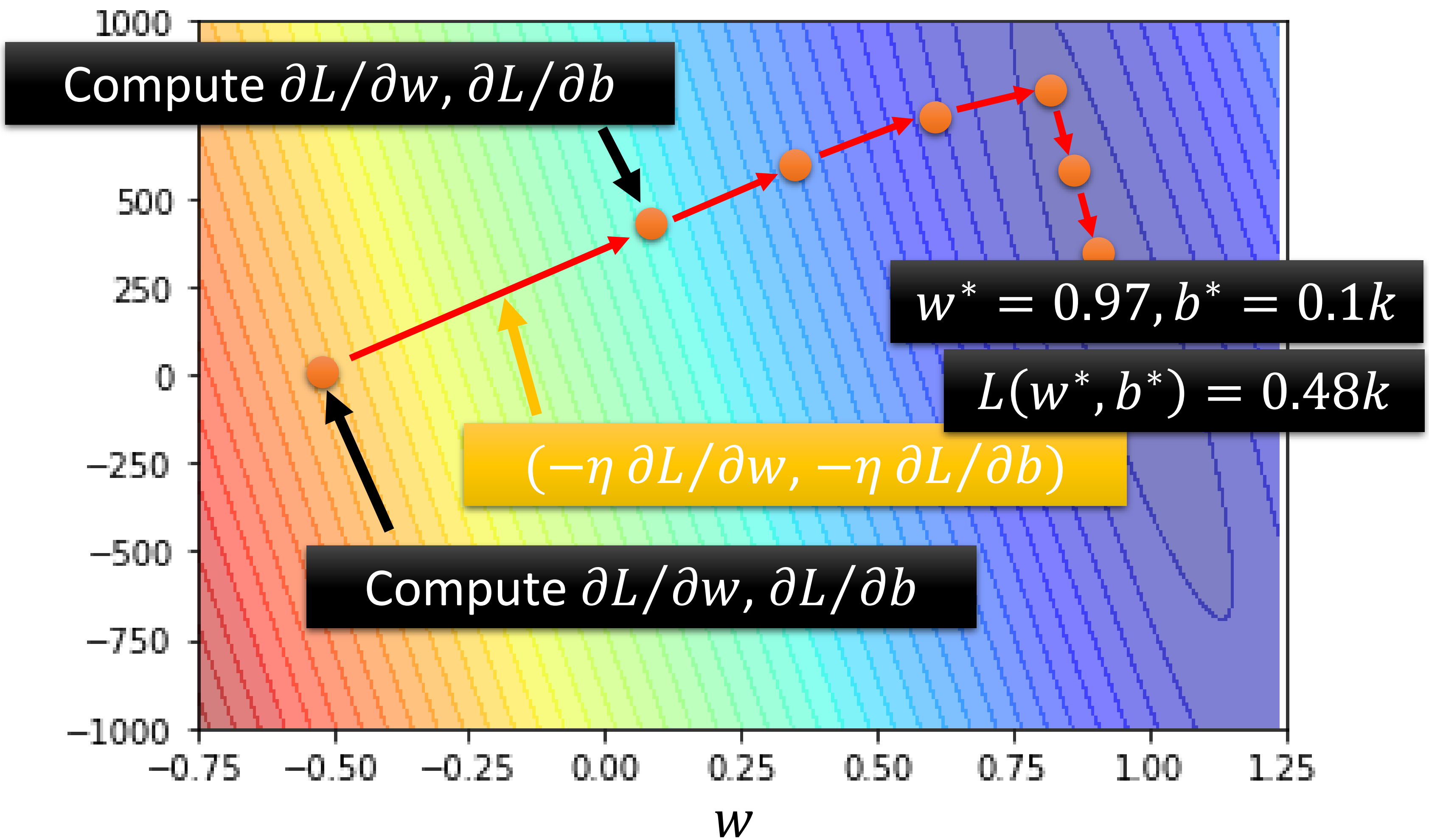
Can be done in one line in most deep learning frameworks

- Update w and b interatively

Gradient Descent

Model $y = b + wx_1$

$$w^*, b^* = arg\min_{w,b} L$$

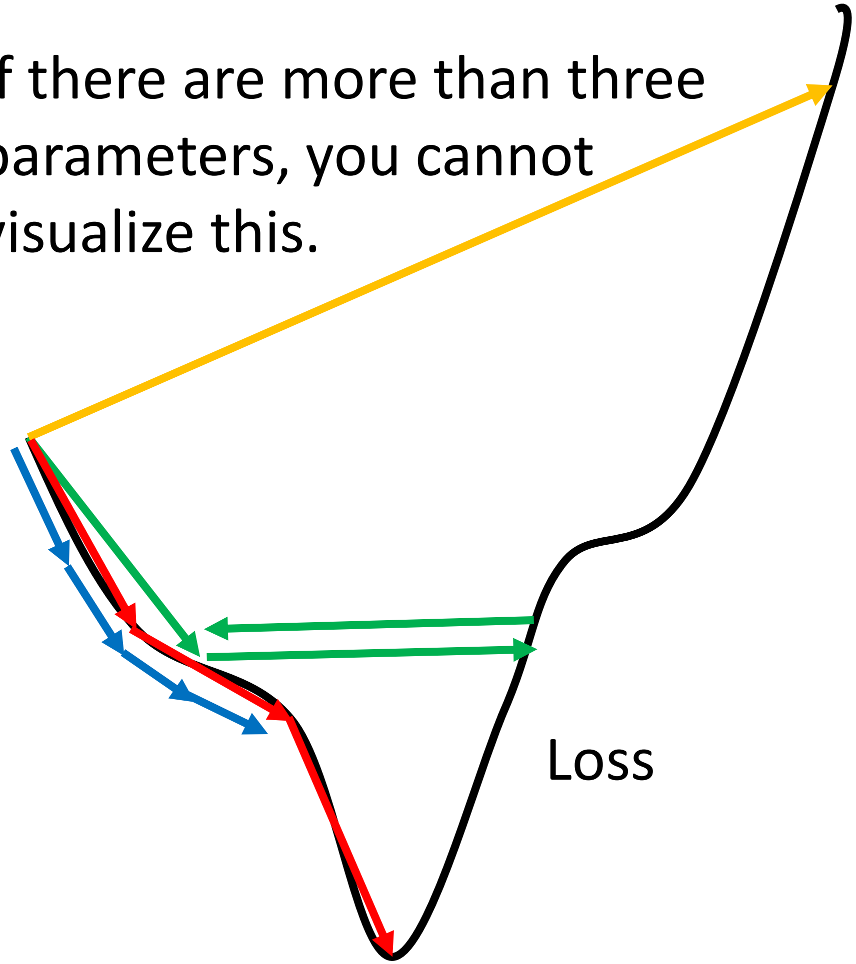


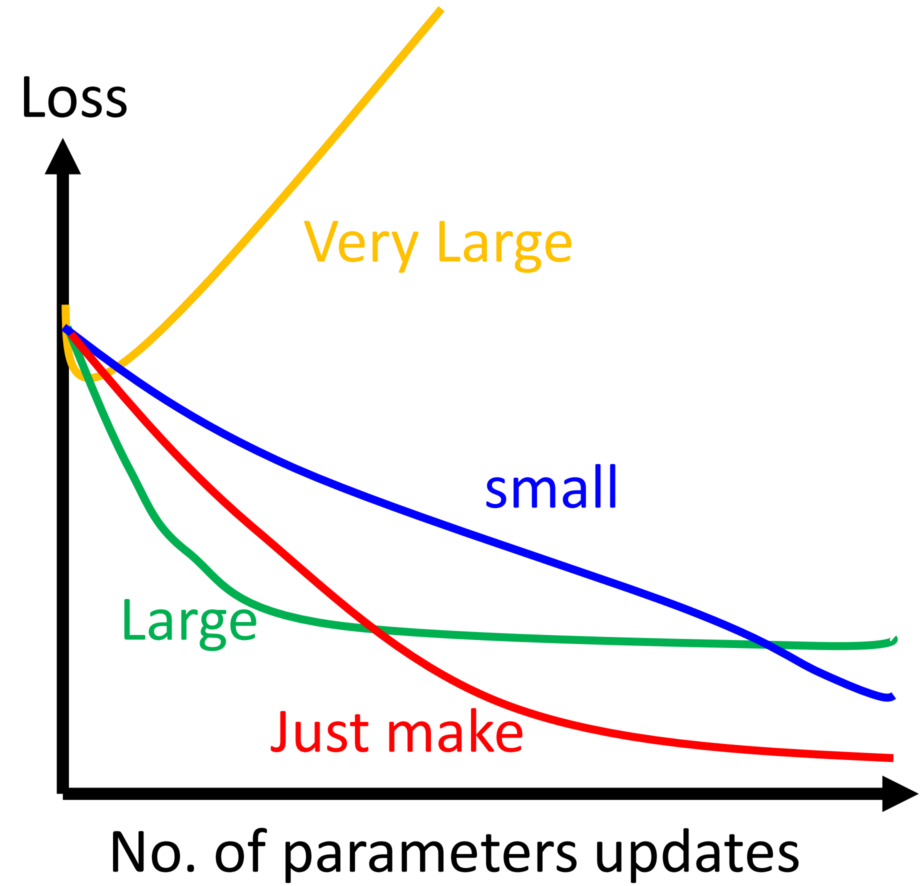
Learning Rate

$$\theta^{i} = \theta^{i-1} - \eta \nabla L\left(\theta^{i-1}\right)$$

Set the learning rate η carefully

If there are more than three parameters, you cannot visualize this.





But you can always visualize this.

Gradient Descent

Tip 1: Adaptive Learning Rate

Adaptive LR

Adagrad

$$w^{t+1} \leftarrow w^t - \frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}} g^t$$

- Popular & Simple Idea: Reduce the learning rate by some factor every few epochs.
  - At the beginning, we are far from the destination, so we use larger learning rate
  - After several epochs, we are close to the destination, so we reduce the learning rate
  - E.g. 1/t decay: $\eta^t = \eta / \sqrt{t+1}$
- Learning rate cannot be one-size-fits-all
  - Giving different parameters different learning rates

Gradient Descent

Tip 2: Stochastic Gradient Descent

Stochastic Gradient Descent (SGD)

$$L = \sum_n \left(\hat{y}^n - \left(b + \sum w_i x_i^n \right) \right)^2$$

Loss is the summation over all training examples

- ***Gradient Descent*** $\theta^i = \theta^{i-1} - \eta \nabla L\left(\theta^{i-1}\right)$

- ***Stochastic Gradient Descent*** Faster!

Pick an example x^n

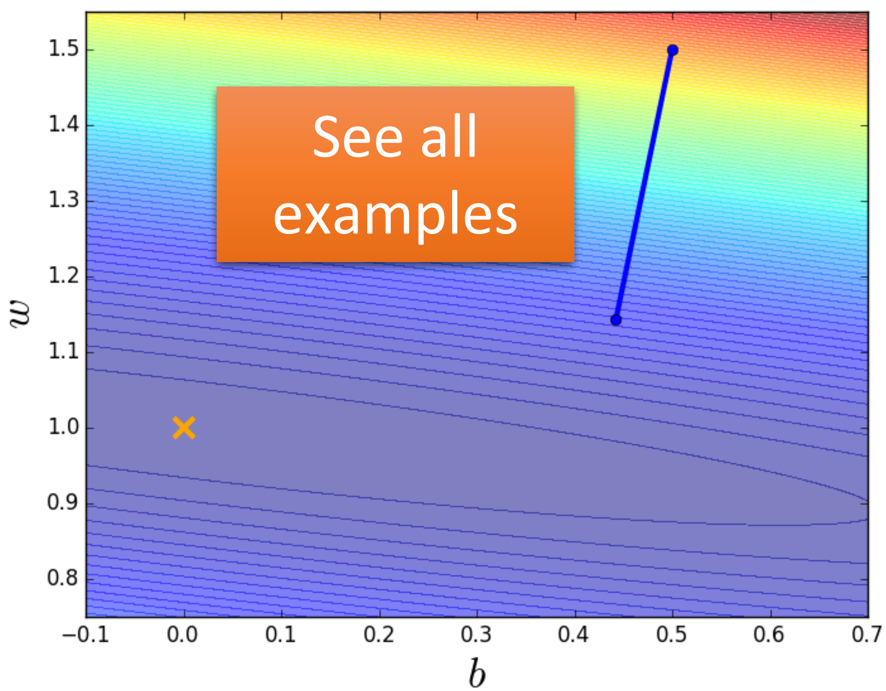
Loss for only one example

$$L^n = \left(\hat{y}^n - \left(b + \sum w_i x_i^n \right) \right)^2 \qquad \theta^i = \theta^{i-1} - \eta \nabla L^n\left(\theta^{i-1}\right)$$

SGD

Gradient Descent

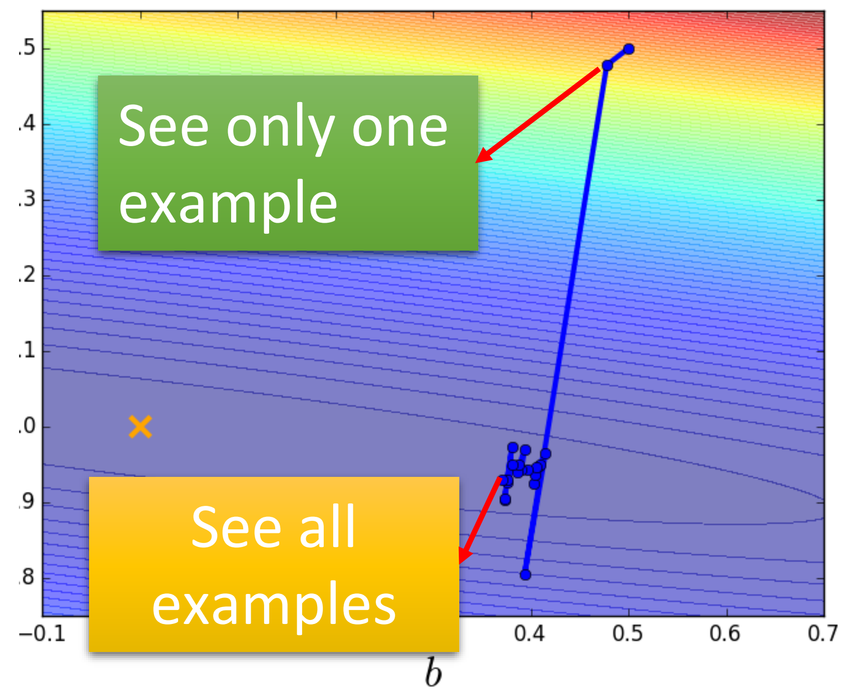
Update after seeing all examples



Stochastic Gradient Descent

Update for each example

If there are 20 examples, 20 times faster.

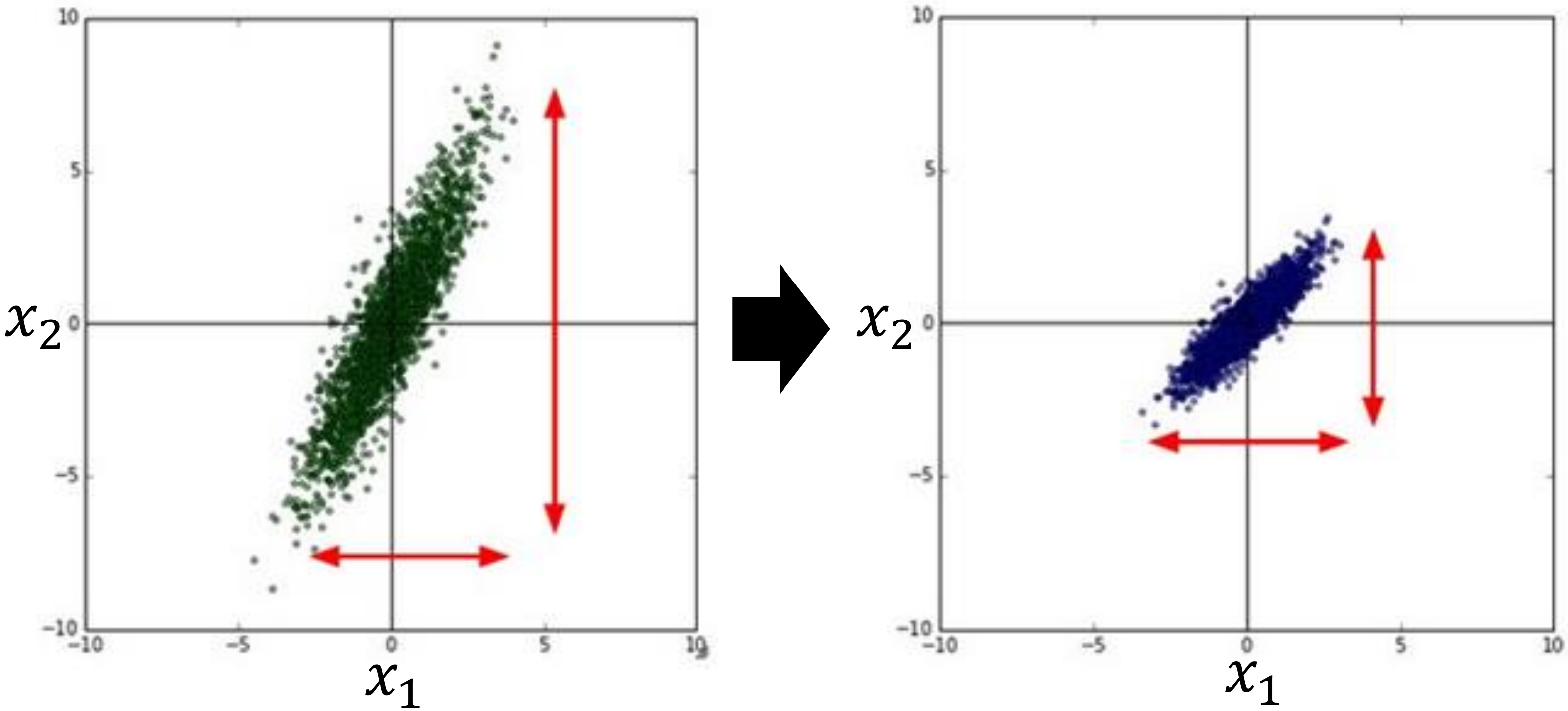


Gradient Descent

Tip 3: Feature Scaling

Feature Scaling

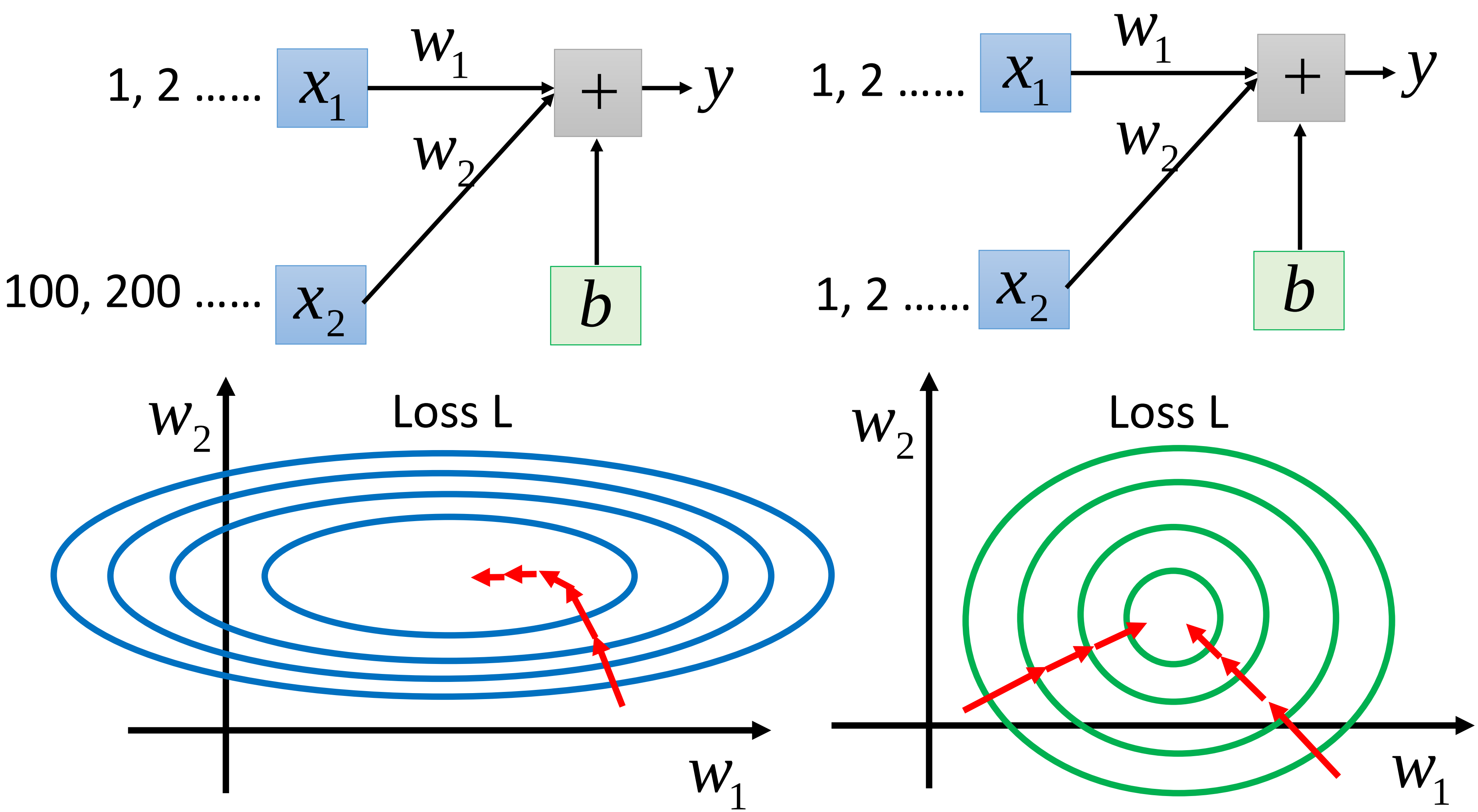
$$y = b + w_1 x_1 + w_2 x_2$$



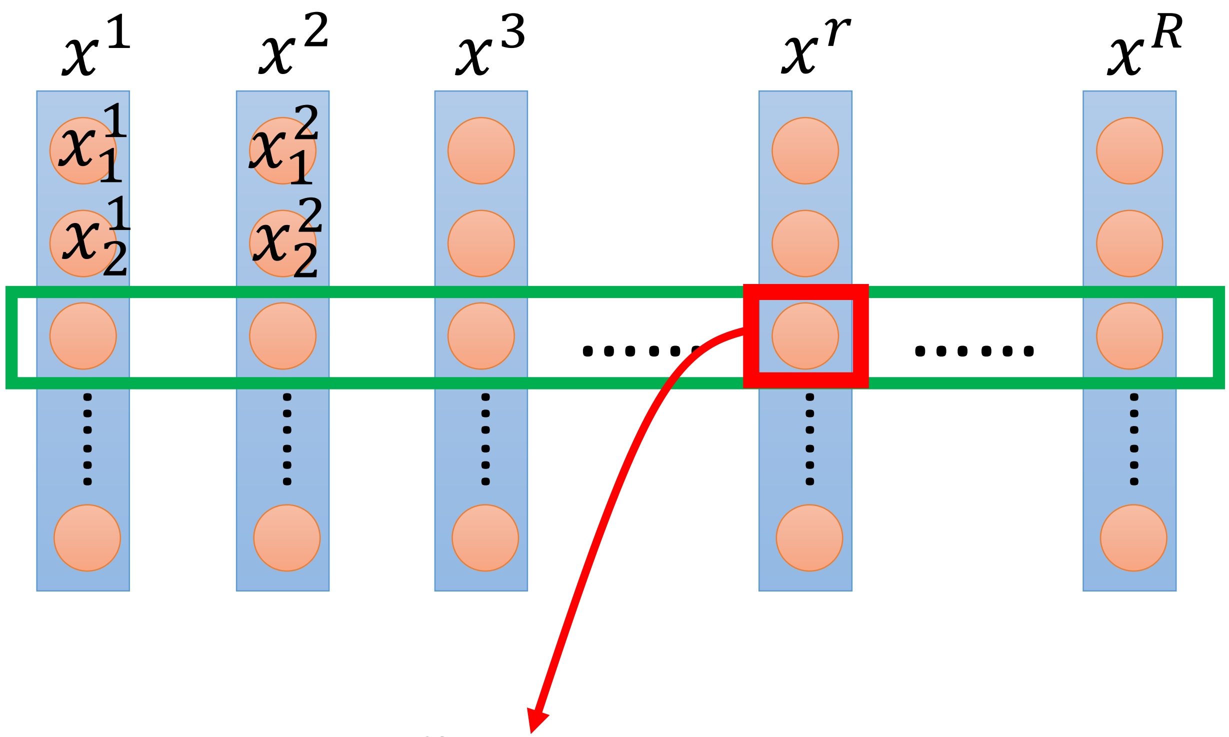
Make different features have the same scaling

Feature Scaling

$y = b + w_1 x_1 + w_2 x_2$



Feature Scaling



For each dimension i:

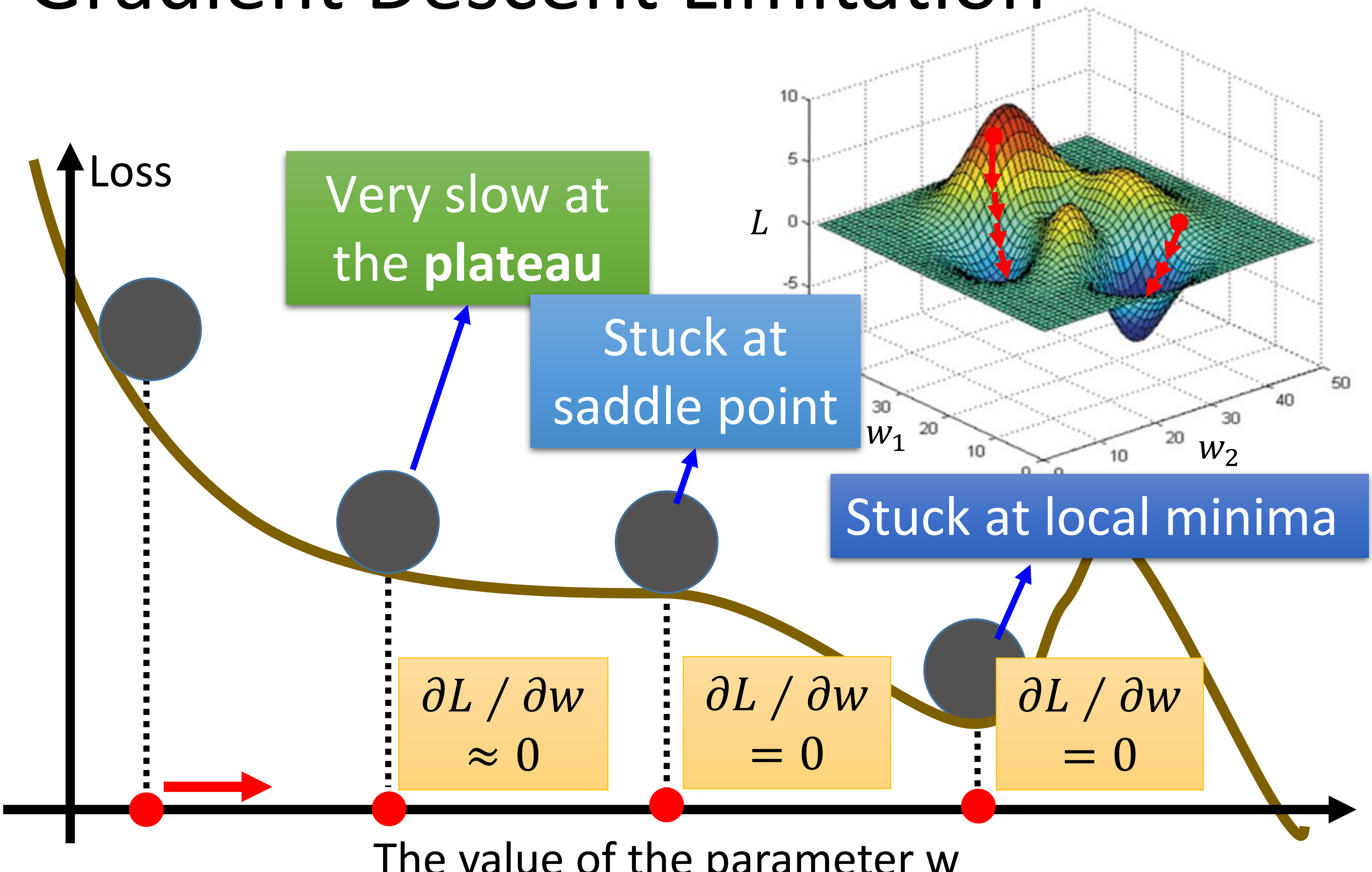
mean: m_i

standard deviation: σ_i

$$x_i^r \leftarrow \frac{x_i^r - m_i}{\sigma_i}$$

The means of all dimensions are 0, and the variances are all 1

Gradient Descent Limitation



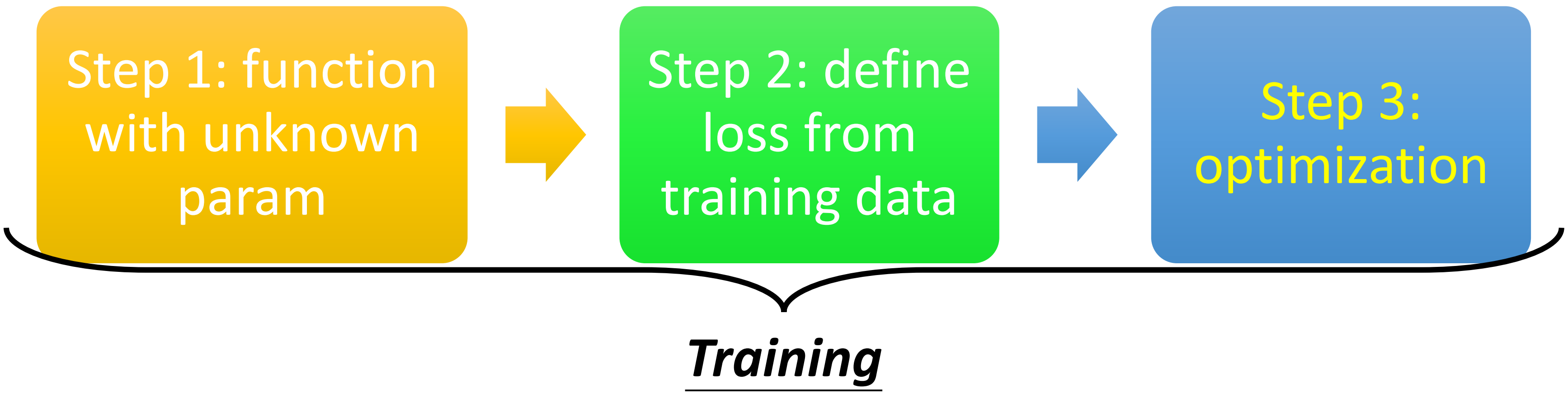
So far, we’ve got optimization

Let’s go back to the machine learning framework.

Machine Learning is so simple ……

$w^* = 0.97, b^* = 0.1k$

$y = b + wx_1$ $\qquad$ $L(w^*, b^*) = 0.48k$



$y = 0.1k + 0.97x_1$ achieves the smallest loss $L = 0.48k$ on data of 2017 – 2020 (**training data**)

How about data of 2021 (**unseen during training**)?

$L' = 0.58k$

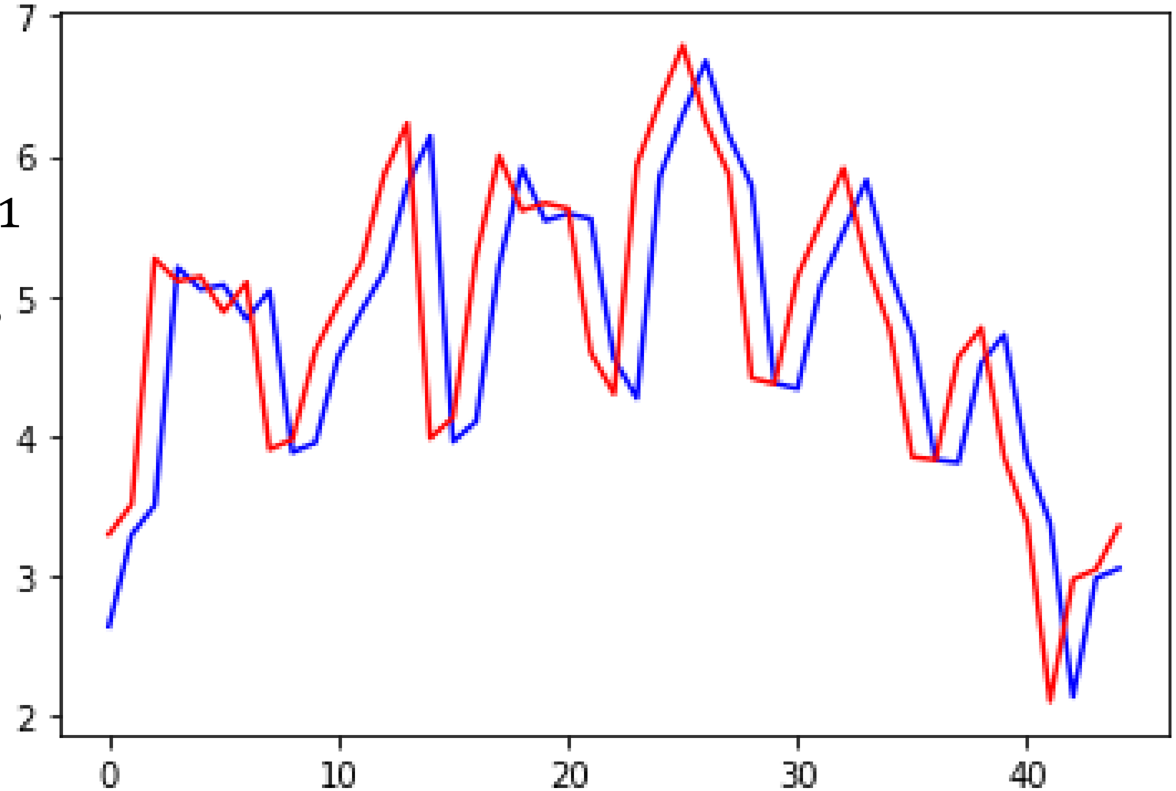
The result

Red: real no. of views

blue: estimated no. of views

$$y = 0.1k + 0.97x_1$$

Views (k)



2021/01/01

2021/02/14

Linear Regression Summary

Model $y = b + wx_1$

Loss

$e = |y - \hat{y}|$ L is mean absolute error (**MAE**)

$e = (y - \hat{y})^2$ L is mean square error (**MSE**)

Optimization Gradient Descent

Linear Regression Summary

- **Strength & Weakness**

✓Easy to understand and implement

✓Good comprehensibility

✕Performs poorly when there are non-linear relationships

✕Not flexible enough to capture more complex patterns

Today's Topics

- Regression
- Linear Regression
- Gradient Descent
- ***Advanced Regression Methods***

Regularization

- Complex model leads to overfitting. Regularization is a way to mitigate this undesirable behavior.
- Through regularization, we can *penalize* complex models and favor simpler ones.

$$min_w \ \mathcal{L}(w) + \Omega(w)$$

- The second term Ω is a regularizer, measuring the complexity of the model given by w.

Ridge Regression

- **Ridge Regression with L2 Regularization**

$$\Omega(w) = \lambda \|w\|_2^2$$

$$\boldsymbol{where}\ \|w\|_2^2 = \Sigma_i w_i^2$$

Here the main effect is that large model weights $\boldsymbol{w_i}$ will be **penalized** (avoided), since we consider them "unlikely", while small ones are ok.

When $\boldsymbol{L(w)}$ is **MSE**:

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \sum_{n=1}^{N} \left[y_n - \mathbf{x}_n^\top \mathbf{w} \right]^2 + \lambda \|\mathbf{w}\|_2^2$$

LASSO Regression

- **LASSO Regression with L1 Regularization**

$$\Omega(w) = \lambda \|w\|$$

$$\textbf{\textit{where}}\ \|w\|_1 = \Sigma_i |w_i|$$

For the L1-regularization the optimum solution is likely going to be **sparse** (only has few non-zero components) compared to the case where we use L2-regularization.

When $\boldsymbol{L(w)}$ is **MSE**:

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \sum_{n=1}^{N} [y_n - \mathbf{x}_n^\top \mathbf{w}]^2 \ + \ \lambda \|\mathbf{w}\|_1$$

L1 VS L2

- **Ridge Regression** tends to distribute weights evenly among related features
- **Lasso Regression** tends to select one from the relevant features, and the rest of the feature weights decay to zero (*Feature Selection*)

Summary

- **Regression**
  - difference with classification
- **Linear Regression**
  - model, loss and optimization
- **Gradient Descent**
  - steps, learning rate, …
- **Advanced Regression Methods**
  - Ridge
  - Lasso

Some questions…

- Does local minima truly cause the problem?
- How does learning rate η influence the optimization?