

Advanced techniques

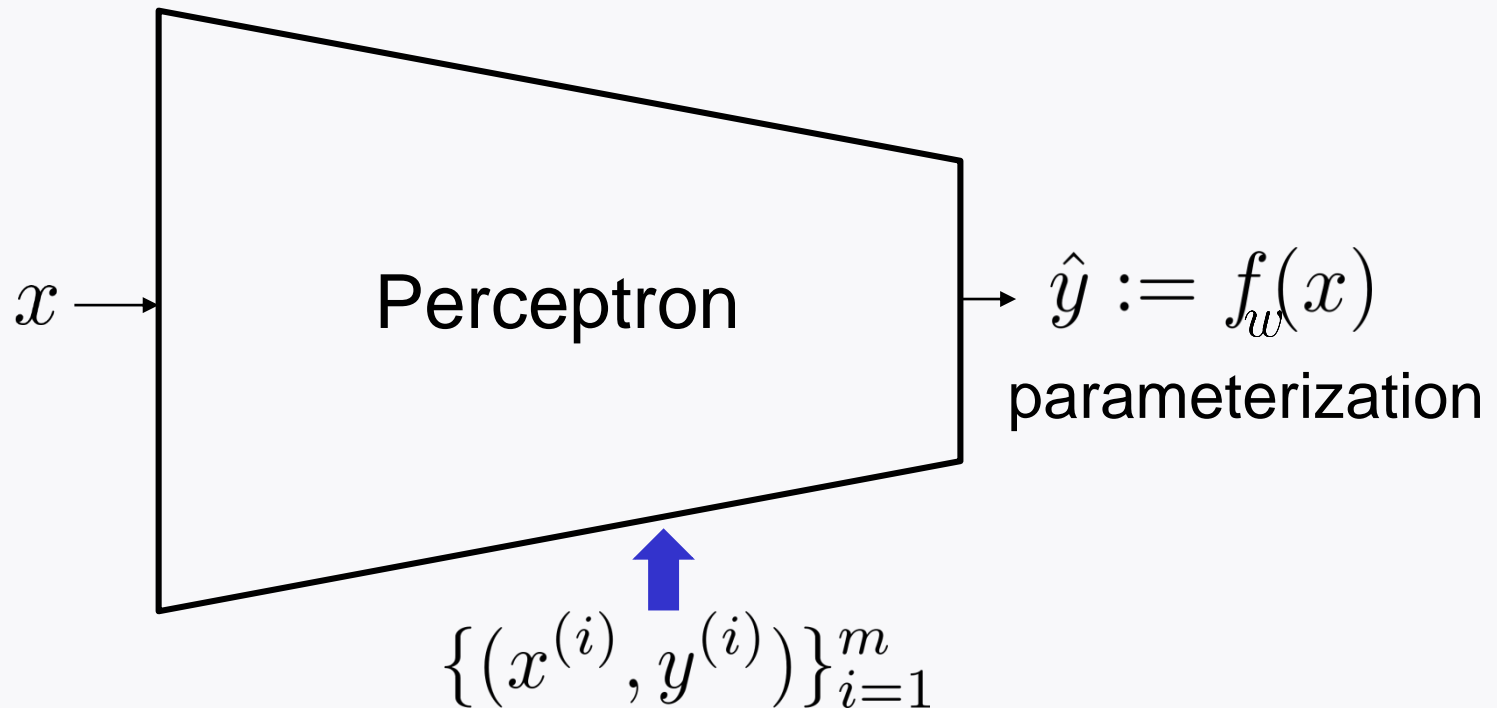
Lecture 4

Changho Suh

January 23, 2024

Data organization & generalization techniques

Recap: Machine learning

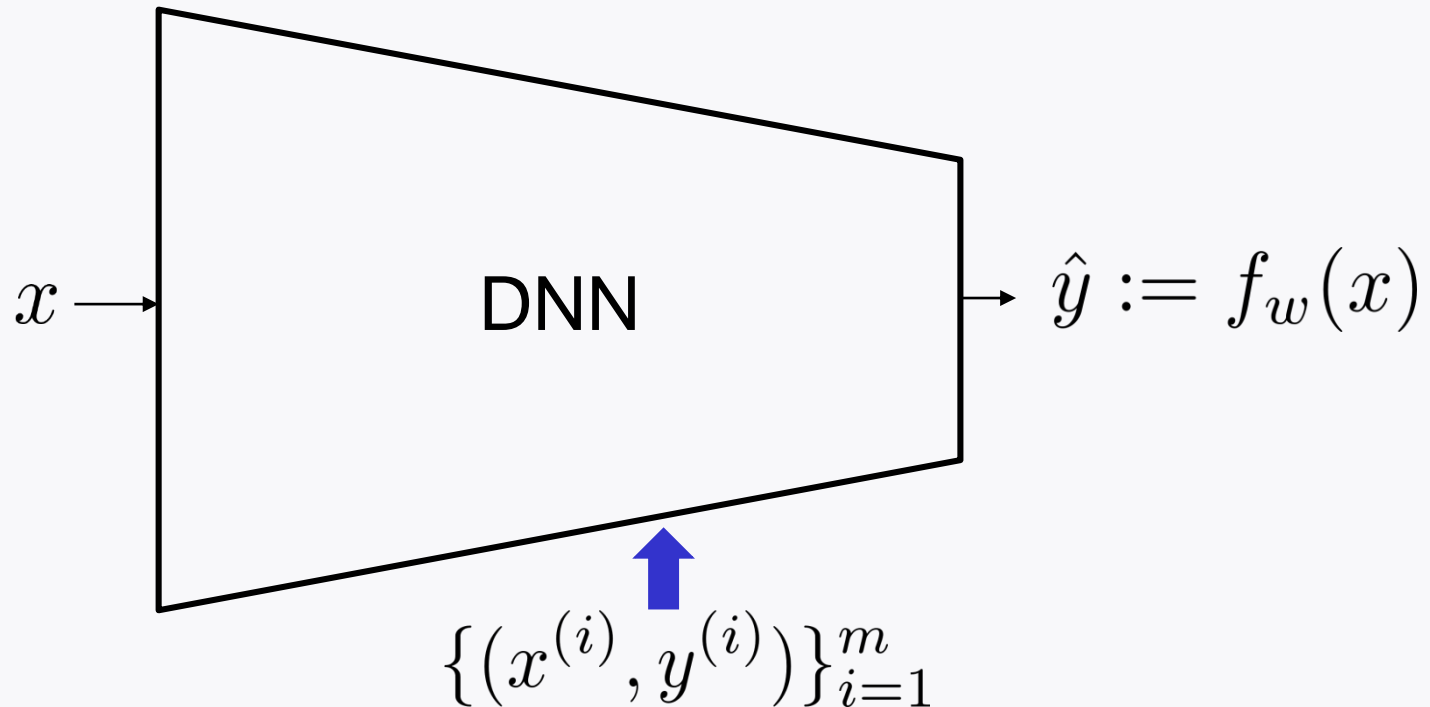


No activation + squared error loss: **Least Squares**

Logistic act. + cross entropy loss: **Logistic regression**

Algorithm: Gradient descent

Recap: Deep neural networks



ReLU (@hidden); **Logistic** (@output); Cross entropy loss

Algorithm: Gradient descent

Efficient method: **backprop**

Practical variant: Adam optimizer

Recap: Scikit-learn coding for LS

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

iris=load_iris()
X=iris.data
y=iris.target
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)

from sklearn.linear_model import RidgeClassifier

Model_LS = RidgeClassifier()
Model_LS.fit(X_train,y_train)
Model_LS.predict(X_test)
Model_LS.score(X_test,y_test)
```

Recap: Scikit-learn coding for LR

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

iris=load_iris()
X=iris.data
y=iris.target
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)

from sklearn.linear_model import LogisticRegression

Model_LR = LogisticRegression()

Model_LR.fit(X_train,y_train)

Model_LR.predict(X_test)

Model_LR.score(X_test,y_test)
```

Recap: TensorFlow coding for DNN

```
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense

(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train, X_test = X_train/255.0, X_test/255.0

Model_NN = Sequential()
Model_NN.add(Flatten(input_shape=(28,28)))
Model_NN.add(Dense(128,activation='relu'))
Model_NN.add(Dense(10,activation='softmax'))

Model_NN.compile(optimizer='adam',
loss='sparse_categorical_crossentropy', metrics=['acc'])
Model_NN.fit(X_train, y_train, epochs=10)
Model_NN.predict(X_test)
Model_NN.evaluate(X_test, y_test)
```

Question

How to improve model performance?

Outline of today's lectures

Will explore several techniques for **improvement**.

1. Data organization (train/validation/test sets)
2. Generalization techniques
3. Weight initialization
4. Techniques for training stability
5. Hyperparameter search
6. Cross validation

Focus of Lecture 4

Will explore several techniques for **improvement**.

1. Data organization (train/validation/test sets)
2. Generalization techniques
3. Weight initialization
4. Techniques for training stability
5. Hyperparameter search
6. Cross validation

Train vs. validation vs. test sets

Data **seen** during training:

train set

validation set

Role: training model
parameters

Role: *Hyperparameter* search

Data **unseen** during training: **test** set

How to split **train/val/test** sets?

Two important factors to consider:

1. How big “ m ” is
2. Data distribution

How big “ m ” is

A deciding factor for the **split ratio**.

Small: $m \leq 1,000$ train:val:test = 60:20:20

Middle: $1,000 \leq m \leq 10,000$ 80:10:10

Large: $10,000 \leq m \leq 1,000,000$ 98:1:1

Ultra-large: $m \geq 1,000,000$ 99.9:0.05:0.05

Set the split ratio such that $m_{\text{test}} \approx 100 \sim 1000$

Data distribution

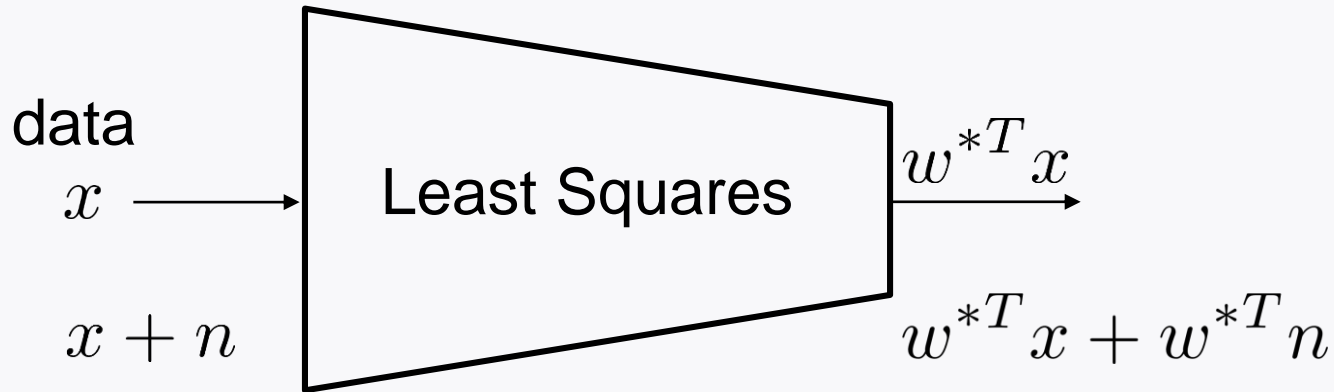
val data dist. ~ **test** data dist. ~ **target** dist.

Take the rest as training data.

Generalization techniques

1. Regularization
2. Data augmentation
3. Early stopping
4. Dropout

Regularization: Motivation



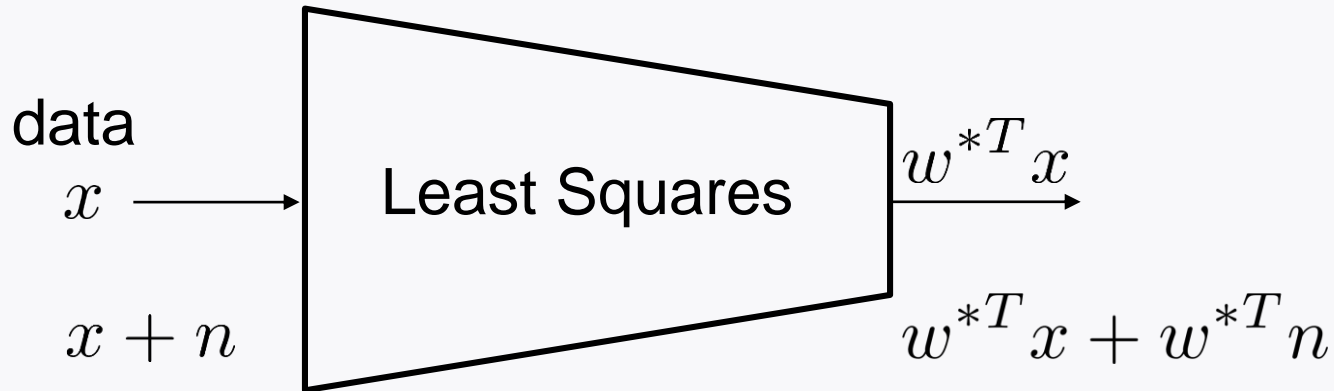
In reality: x contains some **noise**.

Want: model being **robust** to such noise.

Challenge:

Large values of $\|w^*\|$ can **boost up such noise**.

Regularization: Motivation



For robustness, we want: $\|w^*\|^2 \downarrow$

Note: At the same time, we also want:

Loss Function \downarrow

Regularization: Idea

Regulate two objectives at the same time.

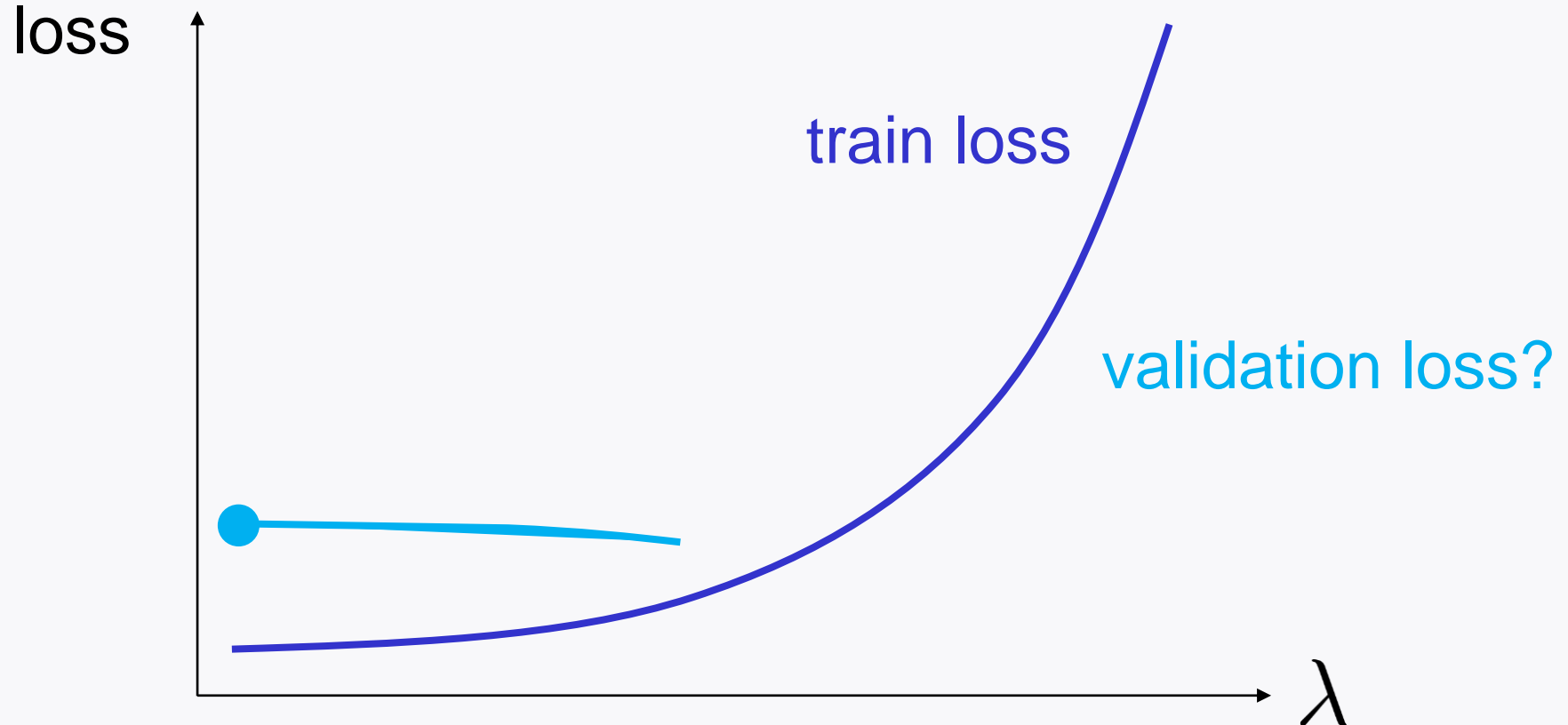
$$\min_w \text{Loss Function} + \lambda \|w\|^2$$

λ : regularization factor

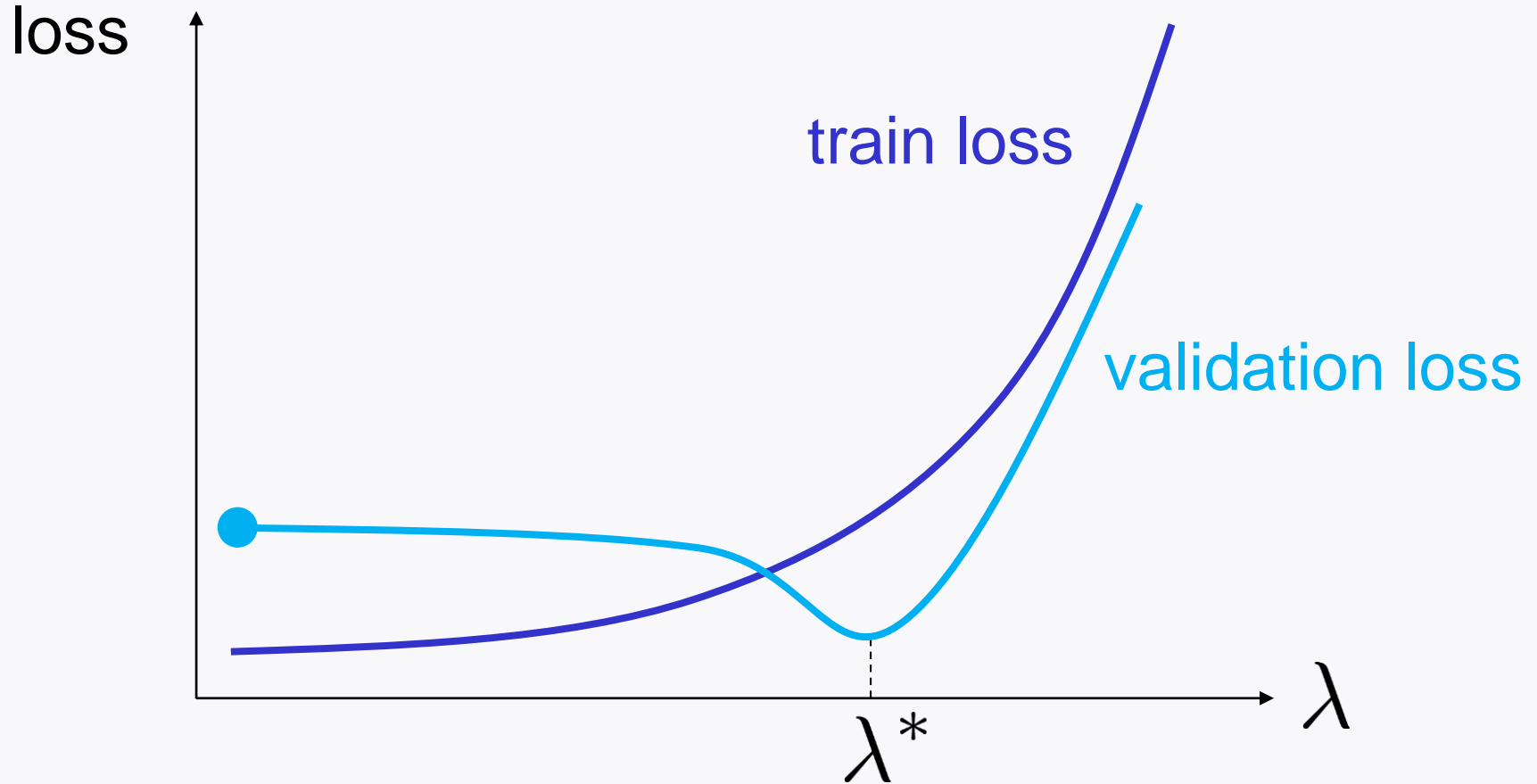
It is a **hyperparameter**!

How to choose?

Regularization: How λ affects?



Regularization: How to choose λ ?



Find the **sweet spot** that minimizes validation loss.

Data augmentation

Idea: Artificially generate diverse data by perturbing original data.

This way: Can make model resilient to **unseen** data.

Hence: Can improve **generalization capability**.

Data augmentation for image data

original



rotation



mirroring



cropping



Data augmentation for numerical data

Original data: $\{(x^{(i)}, y^{(i)})\}_{i=1}^m$

One prominent way is to add random noise:

$$x^{(i)} + n \leftarrow \text{random noise}$$

Look ahead

Will study:

early stopping, dropout

weight initialization

techniques for training stability