# Day 03

Wan Fang
Southern University of Science and Technology

# Day 3

- 8:00 – 9:30 Review of the Initial Ideation
- 9:40 – 11:00 Lecture : Linear Regression and Classification
  Lecture: Neural Networks
- 11:10 – 12:10 Exercise: Build and Play with Neural Networks

- 2:00 – 5:00 Exercise: AI Meets Design Activity II
- 5:00 – 6:00 Review of the day

# AI –> Machine Learning–> Deep Learning

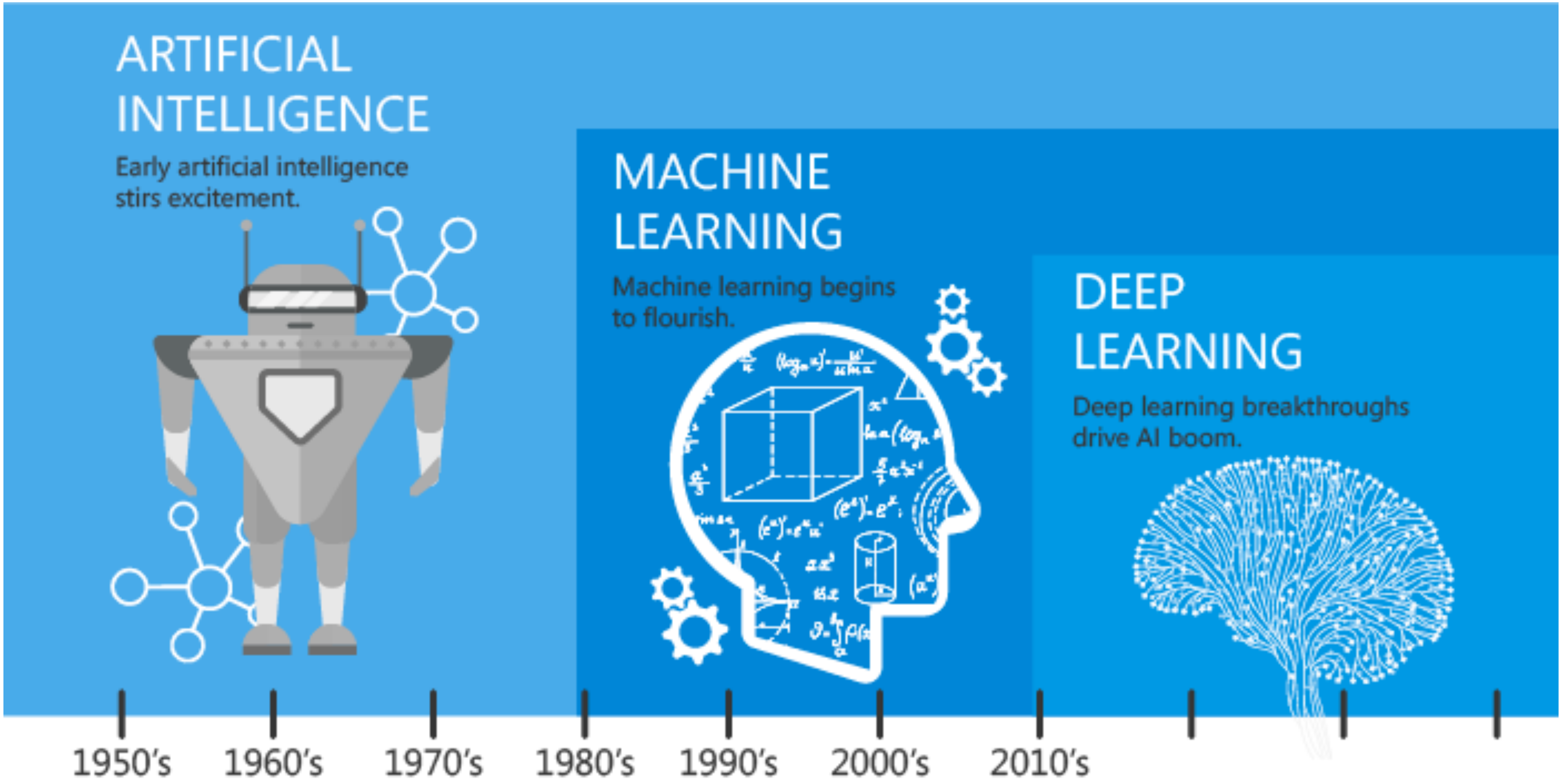
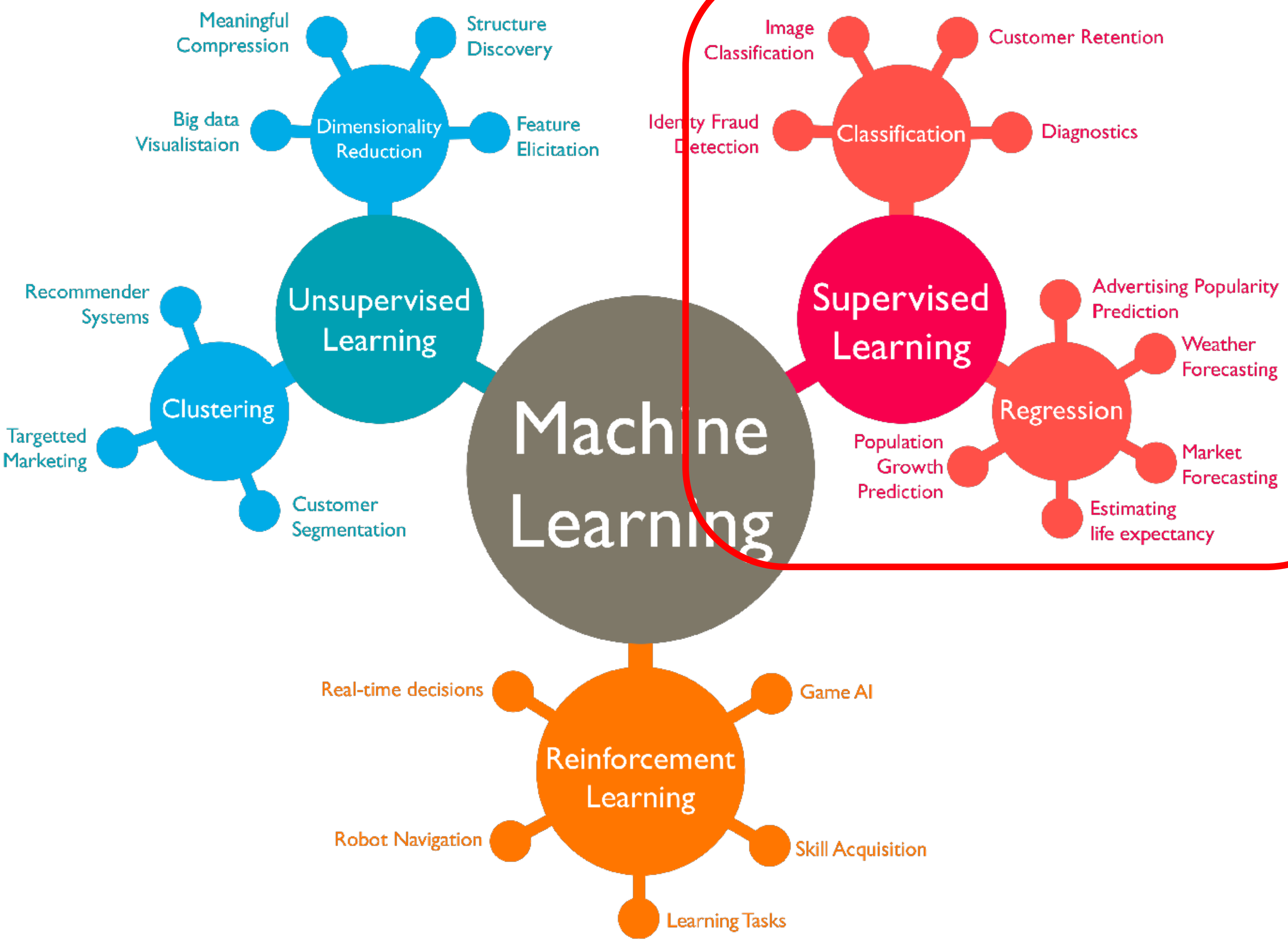


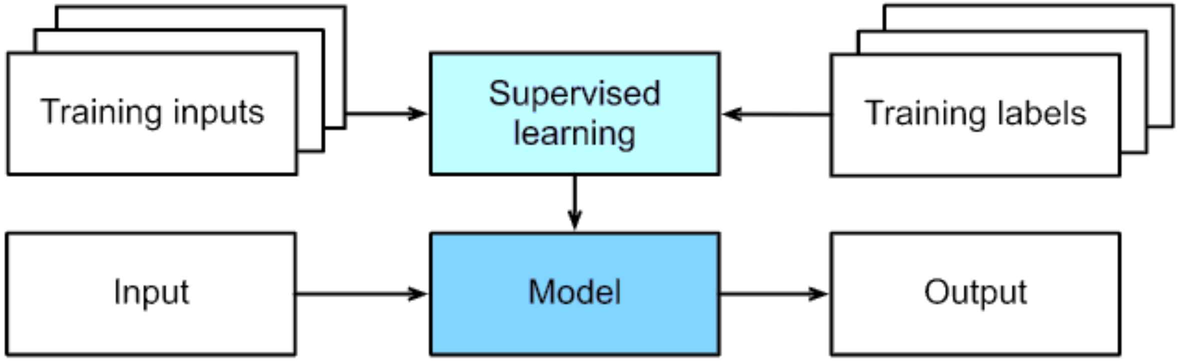

Image: Linked In | Machine Learning vs Deep learning

# Types of Machine Learning?



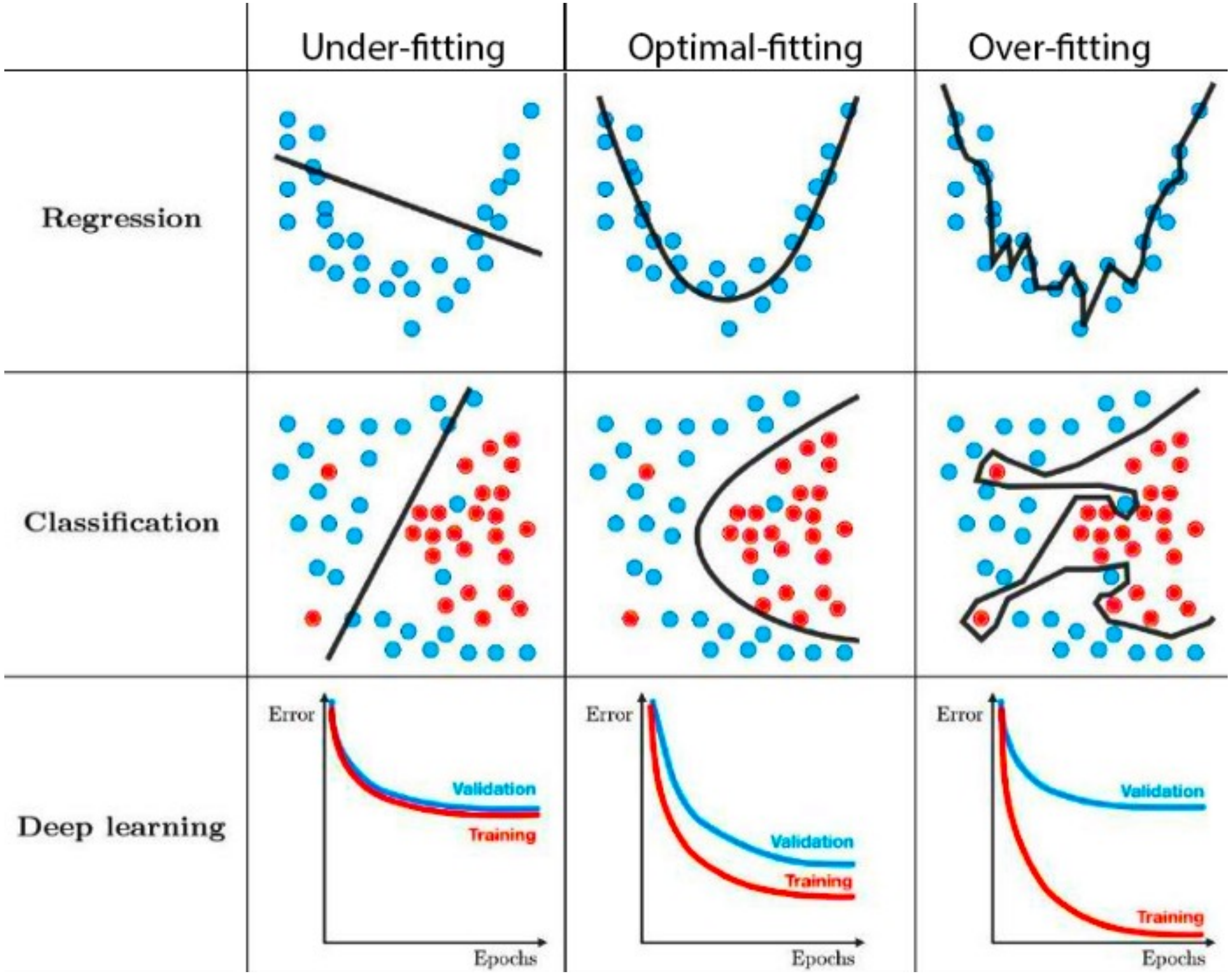
# Supervised Learning

- **Training Data**
  - Data used to learn a model
- **Test Data**
  - Data used to assess the accuracy of model




# Overfitting

- Model performs well on training data but poorly on test data



# Bias and Variance

- **Bias**
  - Expected difference between model's prediction and truth
- **Variance**
  - How much the model differs among training sets

$$\begin{aligned}\text{MSE} &= \mathbb{E}[(\hat{\theta}_m - \theta)^2] \\ &= \text{Bias}(\hat{\theta}_m)^2 + \text{Var}(\hat{\theta}_m)\end{aligned}$$

**Model Scenarios**

- *High Bias*: Model makes inaccurate predictions on training data
- *High Variance*: Model does not generalize to new datasets
- *Low Bias*: Model makes accurate predictions on training data
- *Low Variance*: Model generalizes to new datasets

# Linear Regression

- $\mathbf{w}$
  - The ***weight*** determines the influence of each feature on our prediction, usually a vector form with $w_i$
- $b$
  - The ***bias*** says what value the predicted price should take when all features take 0
- Given a dataset, **our goal** is
  - To choose the weights $\mathbf{w}$ and bias $b$ such that on average, the predictions made based on our model best fit the true prices observed in the data.

$$\hat{y} = w_1 \cdot x_1 + ... + w_d \cdot x_d + b \longrightarrow \hat{y} = \mathbf{w}^T \mathbf{x} + b.$$

# Linear Regression

$$\hat{y}^i = w_1 x_1^i + w_2 x_2^i + \cdots + w_d x_d^i + b$$

index label data point

$i$ $y^i$ $[\, x_1^i$ $x_2^i$ $x_{\dots}^i$ $x_d^i \,]$

| City | Number of weekly riders | Price per week ($) | Population of city | Monthly income of riders ($) | Average parking rates per month ($) |
|---|---|---|---|---|---|
| 1 | 192000 | 15 | 1800000 | 5800 | 50 |
| 2 | 190400 | 15 | 1790000 | 6200 | 50 |
| 3 | 191200 | 15 | 1780000 | 6400 | 60 |
| 4 | 177600 | 25 | 1778000 | 6500 | 60 |
| 5 | 176800 | 25 | 1750000 | 6550 | 60 |
| 6 | 178400 | 25 | 1740000 | 6580 | 70 |
| 7 | 180800 | 25 | 1725000 | 8200 | 75 |
| 8 | 175200 | 30 | 1725000 | 8600 | 75 |
| 9 | 174400 | 30 | 1720000 | 8800 | 75 |
| 10 | 173920 | 30 | 1705000 | 9200 | 80 |
| 11 | 172800 | 30 | 1710000 | 9630 | 80 |
| 12 | 163200 | 40 | 1700000 | 10570 | 80 |
| 13 | 161600 | 40 | 1695000 | 11330 | 85 |
| 14 | 161600 | 40 | 1695000 | 11600 | 100 |
| 15 | 160800 | 40 | 1690000 | 11800 | 105 |
| 16 | 159200 | 40 | 1630000 | 11830 | 105 |
| 17 | 148800 | 65 | 1640000 | 12650 | 105 |
| 18 | 115696 | 102 | 1635000 | 13000 | 110 |
| 19 | 147200 | 75 | 1630000 | 13224 | 125 |
| 20 | 150400 | 75 | 1620000 | 13766 | 130 |
| 21 | 152000 | 75 | 1615000 | 14010 | 150 |
| 22 | 136000 | 80 | 1605000 | 14468 | 155 |
| 23 | 126240 | 86 | 1590000 | 15000 | 165 |
| 24 | 123888 | 98 | 1595000 | 15200 | 175 |
| 25 | 126080 | 87 | 1590000 | 15600 | 175 |
| 26 | 151680 | 77 | 1600000 | 16000 | 190 |
| 27 | 152800 | 63 | 1610000 | 16200 | 200 |

# Linear Regression

$$\hat{y} = w_1 \cdot x_1 + \ldots + w_d \cdot x_d + b \longrightarrow \hat{y} = \mathbf{w}^T \mathbf{x} + b$$

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w} + b$$

- Vectorization
  - All features into a vector **x** for a single data point
  - All weights into a vector **w**
  - Our entire dataset as the *design matrix* **X,** including one row for every example and one column for every feature

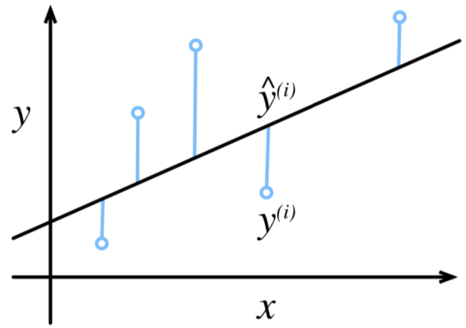
$$\mathbf{X} = \begin{bmatrix} x_1^{(1)} & \cdots & x_d^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(i)} & \cdots & x_d^{(i)} \end{bmatrix}$$

one row for every example

one column
for every feature

# Loss Function

- To quantify the distance between the ***predicted*** and ***real*** value.
  - usually be a non-negative number where smaller values are better
  - perfect predictions incur a loss of 0
- The Sum of Squared Errors $l^{(i)}(\mathbf{w}, b) = \frac{1}{2}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$
  - the empirical error is only
    a function of the model parameters
- Loss Function as an averaged SSE

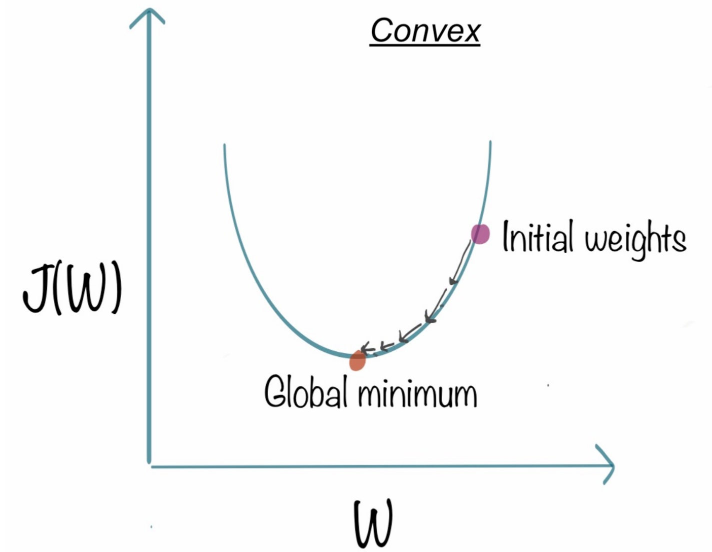


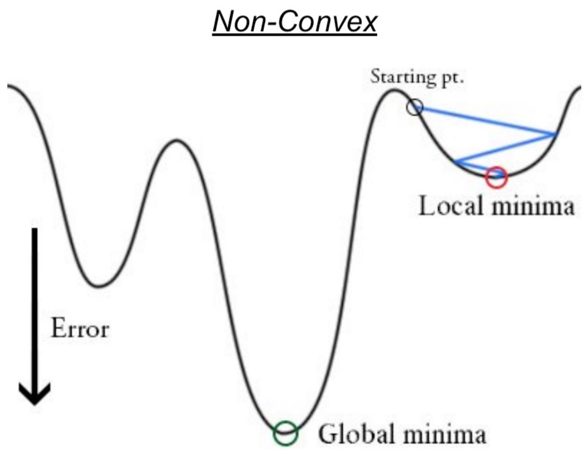

$$L(\mathbf{w}, b) = \frac{1}{n}\sum_{i=1}^{n} l^{(i)}(\mathbf{w}, b) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{2}\left(\mathbf{w}^\top \mathbf{x}^{(i)} + b - y^{(i)}\right)^2$$

$$\mathbf{w}^*, b^* = \underset{\mathbf{w}, b}{\operatorname{argmin}}\ L(\mathbf{w}, b)$$

# Gradient Descent

- **Iteratively reducing** the error by updating the parameters in the direction that incrementally lowers the loss function
  - On *convex* loss surfaces, it will eventually converge to a global minimum
  - For *nonconvex* surfaces, it will at least lead towards a (hopefully good) local minimum.



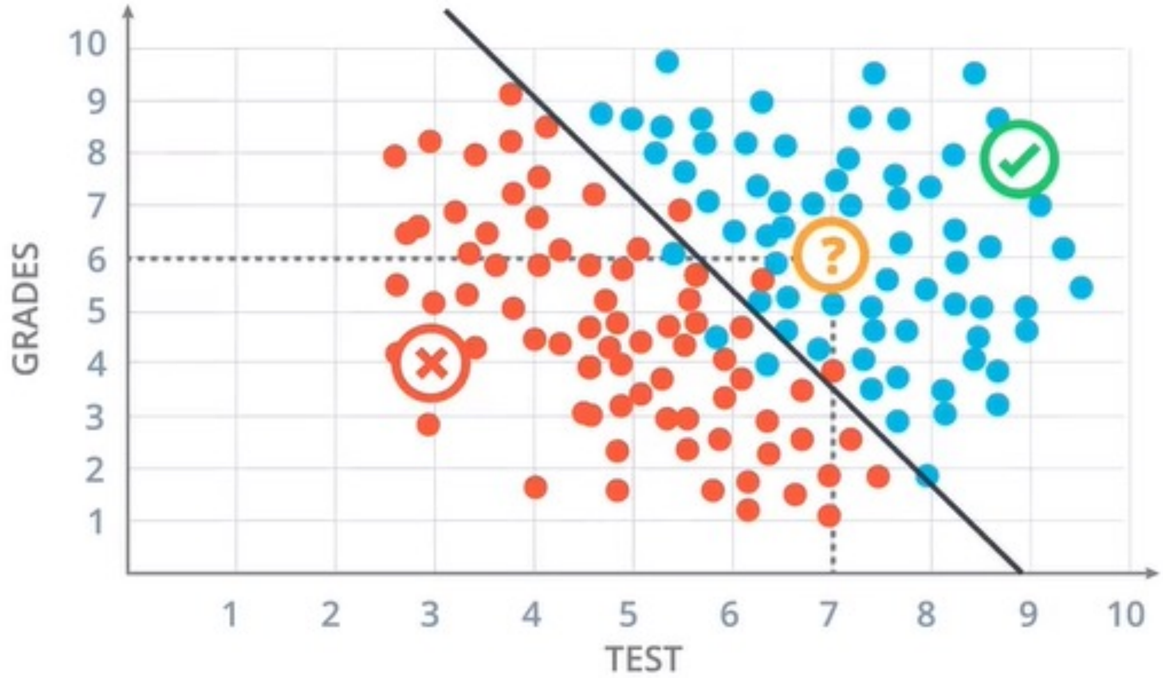




- The key technique for optimizing *nearly any* deep learning model

# Linear Classification

- Hypothesis
  - Acceptance depending on Test and Grade
- Data
  - $\left(x^{(i)}, y^{(i)}\right)$
- Input
  - $x_1^{(i)}$ as test scores and $x_2^{(i)}$ as test scores
- Output
  - $\hat{y}^{(i)}$ as a threshold decision of Accept or Reject
- Model
  - A linear boundary line to separate the data
    - $w_1 x_1 + w_2 x_2 + b = 0$
  - A threshold to activate a decision against the line
    - $> 0$: Accept; $< 0$: Reject

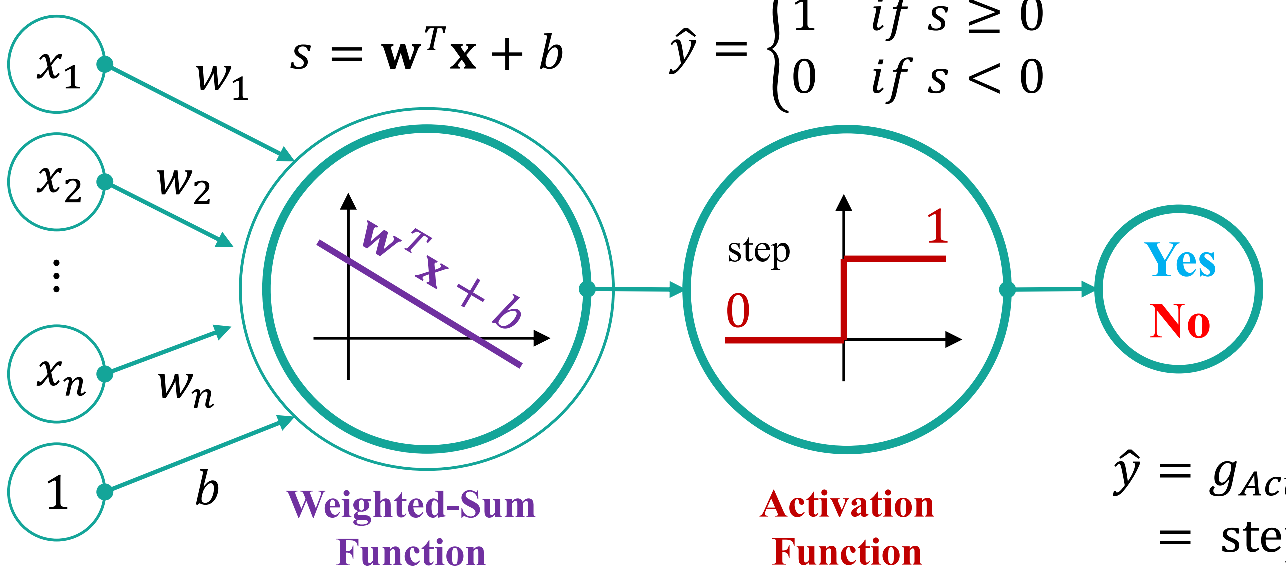
*An example of acceptance at a University*




A Linear Boundary Line of $2x_1 + x_2 - 18 = 0$ as a decision criteria from regression to classification

# Perceptron with an Activation Function

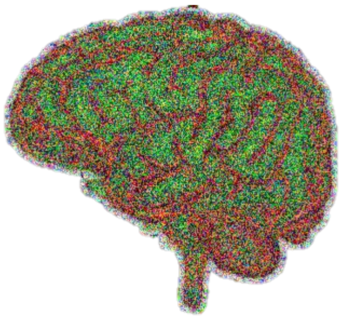
- An Artificial Neuron with two nodes
  - **Weighted-sum node**
    - Calculate a linear equation $s(x)$ with inputs on the weights plus bias
  - **Activation node**
    - Apply the step function to get the predicted result $\hat{y}(s)$

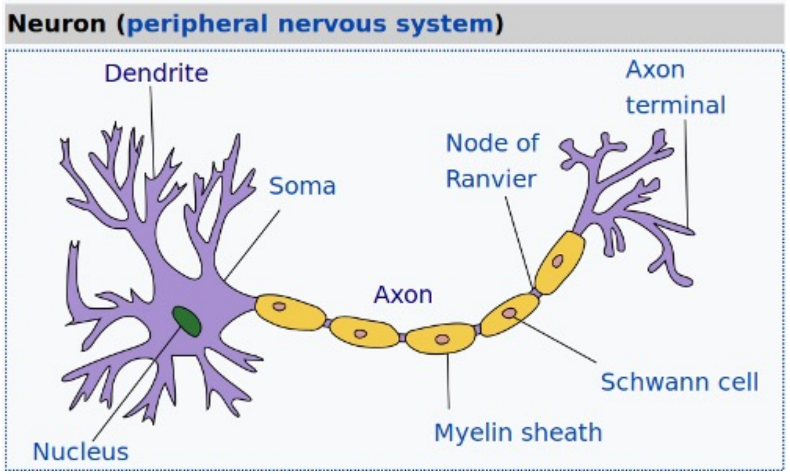


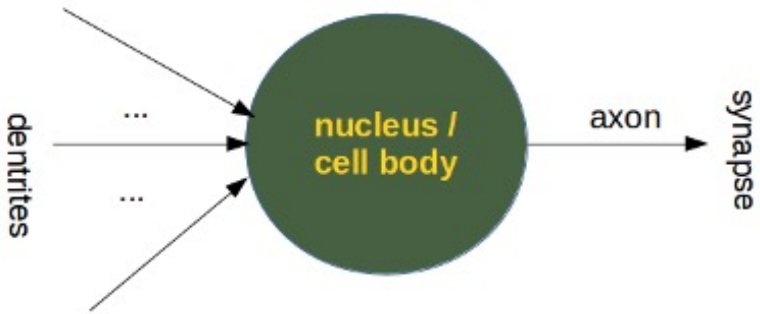

$$\hat{y} = \begin{cases} 1 & if\ s \geq 0 \\ 0 & if\ s < 0 \end{cases}$$

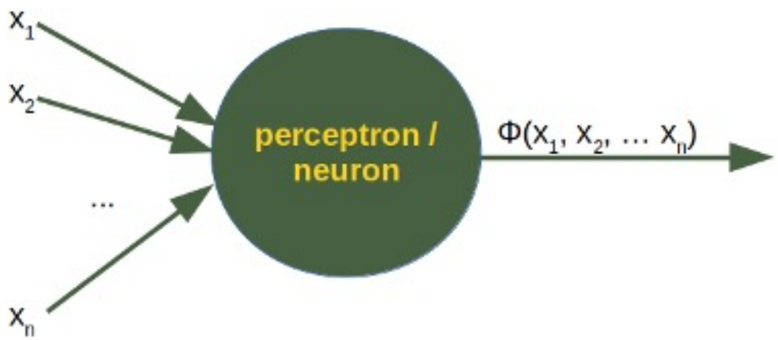
$$\begin{aligned} \hat{y} &= g_{Activation}\left[f_{WeightedSum}(\mathbf{x})\right] \\ &= \mathrm{step}(s, 0) \\ &= \mathrm{step}(\mathbf{w}^T\mathbf{x} + b, 0) \end{aligned}$$

# A Perceptron as an Artificial Neuron





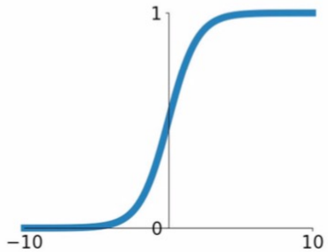



# Activation Function

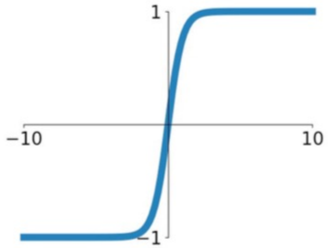
**Sigmoid**

$\sigma(x) = \frac{1}{1+e^{-x}}$



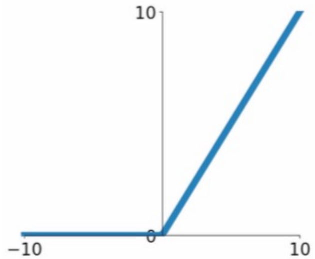

**tanh**

$\tanh(x)$



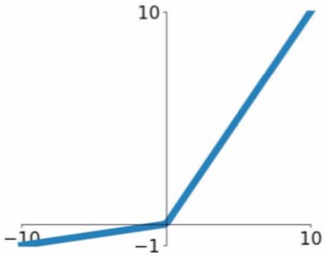

**ReLU**

$\max(0, x)$




**Leaky ReLU**

$\max(0.1x, x)$

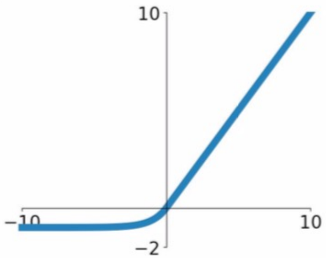



**Maxout**

$\max(w_1^T x + b_1, w_2^T x + b_2)$

**ELU**

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

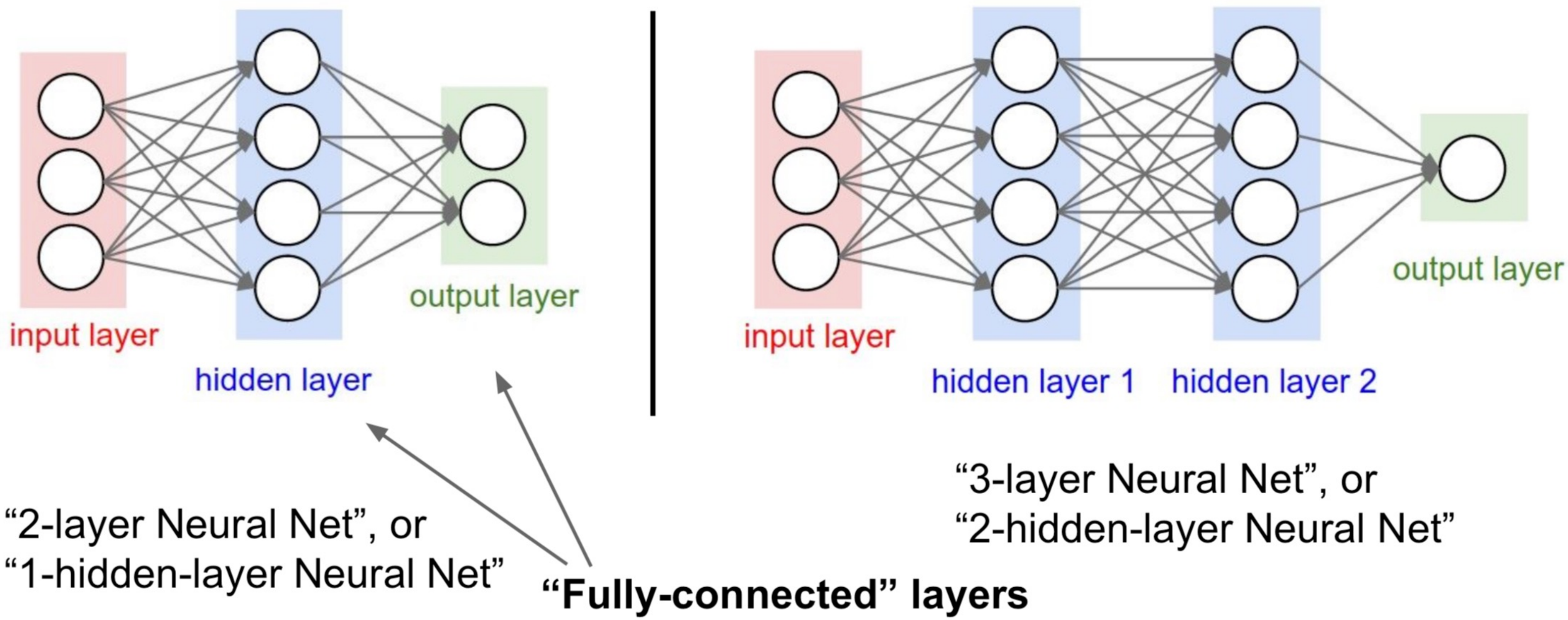


# Limitation of Single Perceptron

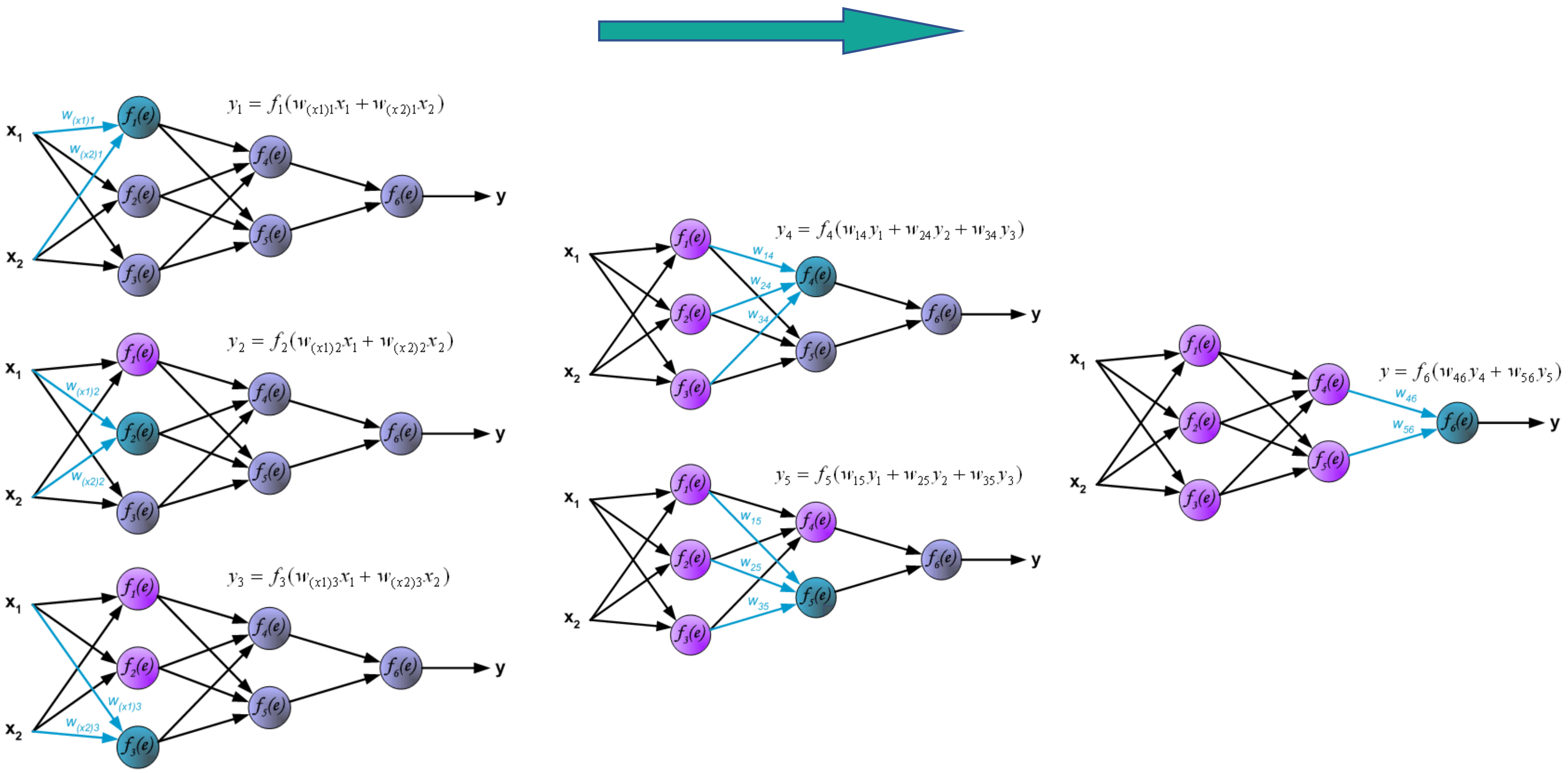
*What if the boundary line is non-linear?*


- Unable to classify nonlinear scenarios

# Multi-Layer Perceptrons



# Forward Propogation




http://galaxy.agh.edu.pl/%7Evlsi/AI/backp_t_en/backprop.html

# Exercise I

- Build your first neural networks on the website
  - https://playground.tensorflow.org/

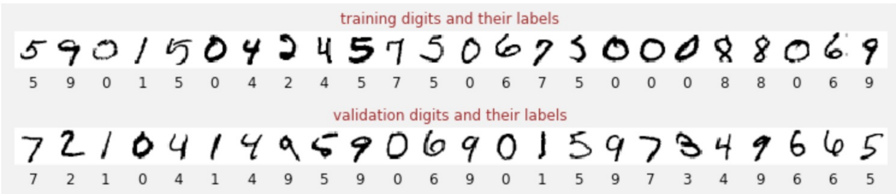
- Play with different data types, features, network structures. Can Neural networks separate nonlinear features?

- Try different hyperparameters.

# Exercise II

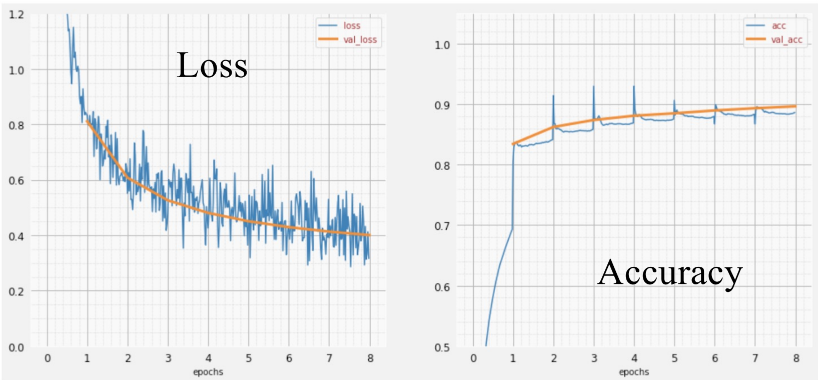

MNIST = Mixed National Institute of Standards and Technology - Download the dataset at http://yann.lecun.com/exdb/mnist/

# A Toy Example of Training a Neural Network




**Tensors** as matrices storage of data

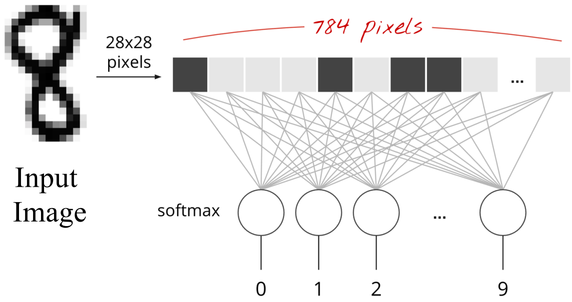
- [28, 28, 1]: A 28x28 pixel grayscale image (Gray)
- [128, 28, 28, 3]: A batch of 128 color images (RGB)




```
468/468 [==============================] - 2s 5ms/step - loss: 0.4290 - acc: 0.8862 - val_loss: 0.4003 - val_acc: 0.8963
Epoch 9/10
378/468 [=======================>......] - ETA: 0s - loss: 0.4230 - acc: 0.8866
```

# A Single-layer Network of Image Classification

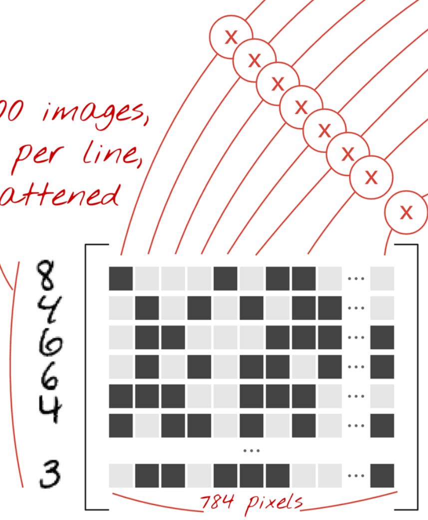
Straightened into an array of features ($x$)




Input Image

Through Softmax operation to predict 10 labels ($y$) with a normalized distrution

Vectorized dataset into a **batch** (of 100 images) for efficient operation

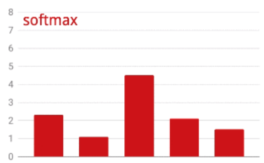



$$W = \begin{bmatrix} w_{0,0} & w_{0,1} & w_{0,2} & w_{0,3} & \cdots & w_{0,9} \\ w_{1,0} & w_{1,1} & w_{1,2} & w_{1,3} & \cdots & w_{1,9} \\ w_{2,0} & w_{2,1} & w_{2,2} & w_{2,3} & \cdots & w_{2,9} \\ w_{3,0} & w_{3,1} & w_{3,2} & w_{3,3} & \cdots & w_{3,9} \\ w_{4,0} & w_{4,1} & w_{4,2} & w_{4,3} & \cdots & w_{4,9} \\ w_{5,0} & w_{5,1} & w_{5,2} & w_{5,3} & \cdots & w_{5,9} \\ w_{6,0} & w_{6,1} & w_{6,2} & w_{6,3} & \cdots & w_{6,9} \\ w_{7,0} & w_{7,1} & w_{7,2} & w_{7,3} & \cdots & w_{7,9} \\ w_{8,0} & w_{8,1} & w_{8,2} & w_{8,3} & \cdots & w_{8,9} \\ & & & \cdots & & \\ w_{783,0} & w_{783,1} & w_{783,2} & & \cdots & w_{783,9} \end{bmatrix}$$

$$\begin{matrix} L_{0,0} & L_{0,1} & L_{0,2} & L_{0,3} & \cdots & L_{0,9} \\ L_{1,0} & L_{1,1} & L_{1,2} & L_{1,3} & \cdots & L_{1,9} \\ L_{2,0} & L_{2,1} & L_{2,2} & L_{2,3} & \cdots & L_{2,9} \\ L_{3,0} & L_{3,1} & L_{3,2} & L_{3,3} & \cdots & L_{3,9} \\ L_{4,0} & L_{4,1} & L_{4,2} & L_{4,3} & \cdots & L_{4,9} \\ & & & \cdots & & \\ L_{99,0} & L_{99,1} & L_{99,2} & & \cdots & L_{99,9} \end{matrix} \quad + \quad b_0\ b_1\ b_2\ b_3\ \cdots\ b_9$$

broadcast

$$L = X.W + b$$




weighted sum of all pixels + bias

$$softmax(L_n) = \frac{e^{L_n}}{||e^L||}$$

neuron outputs

# Softmax on a Batch of Images

Predictions Y[100, 10]    Images X[100, 784]    Weights W[784,10]    Biases b[10]

$$Y = softmax(X.W + b)$$

applied line by line    matrix multiply    broadcast on all lines

tensor shapes in [ ]

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

actual probabilities, "one-hot" encoded

$$-\sum Y_i'.log(Y_i)$$ Cross entropy

computed probabilities    this is a "6"

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| .01 | .03 | .00 | .04 | .03 | .05 | 0.8 | .02 | .01 | .01 |

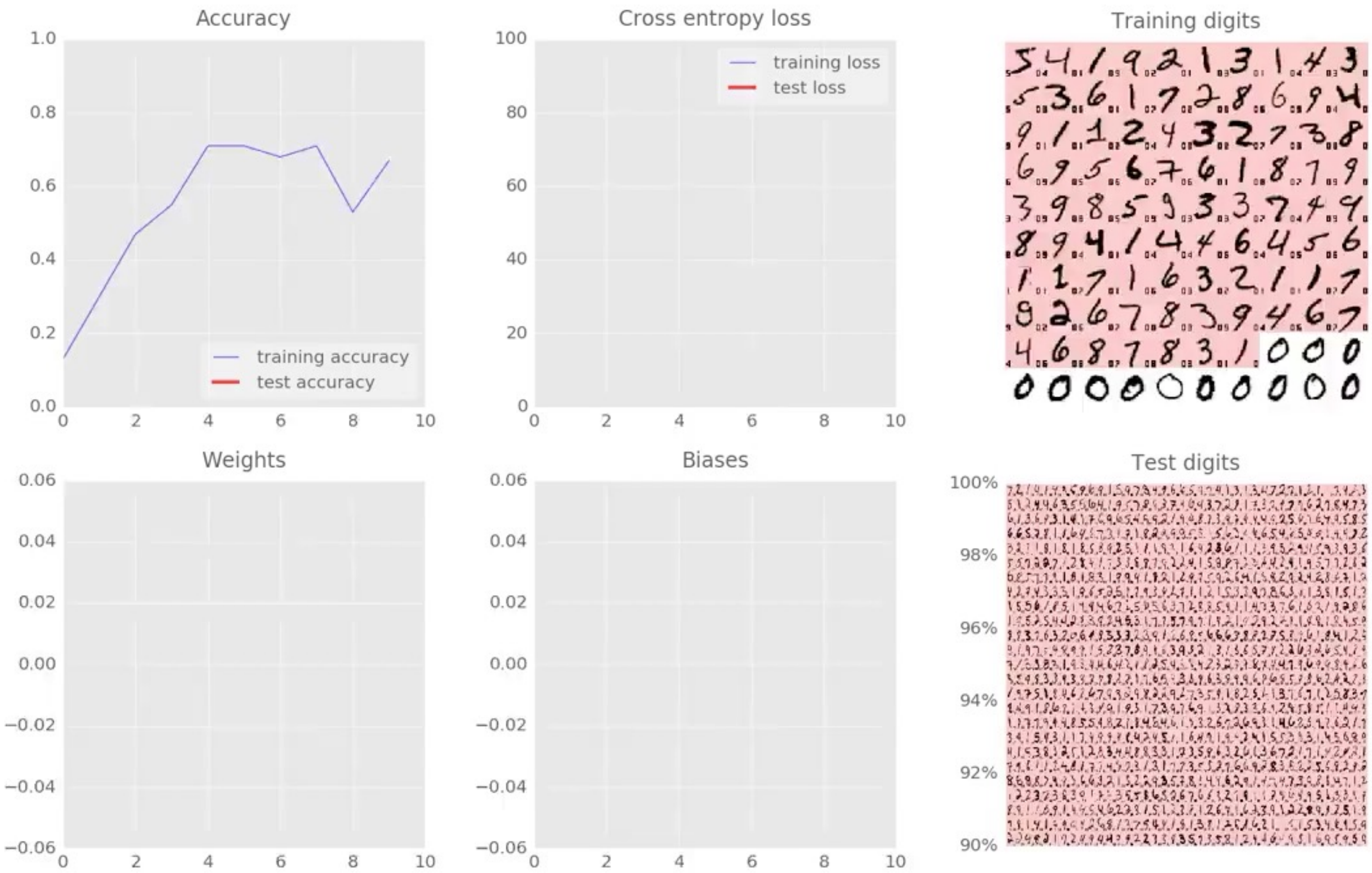
tensor shapes: X[100, 784]    W[748,10]    b[10]

```
Y = tf.nn.softmax(tf.matmul(X, W) + b)
```

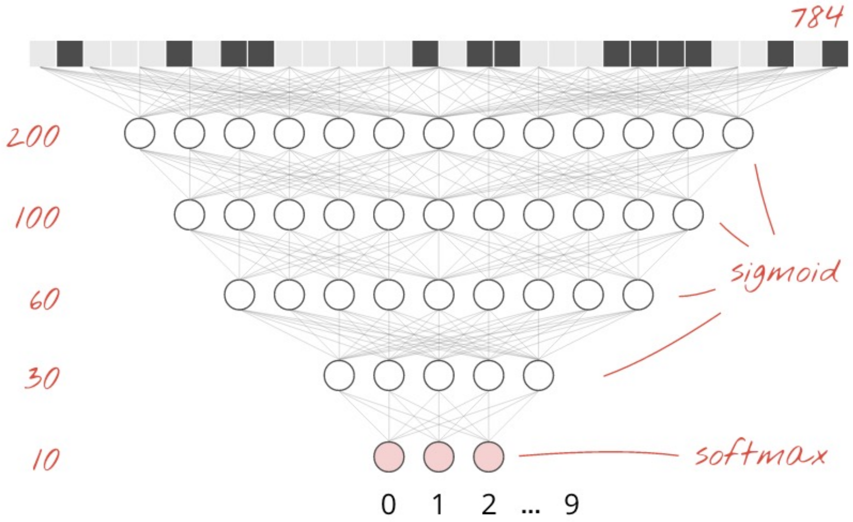
matrix multiply    broadcast on all lines

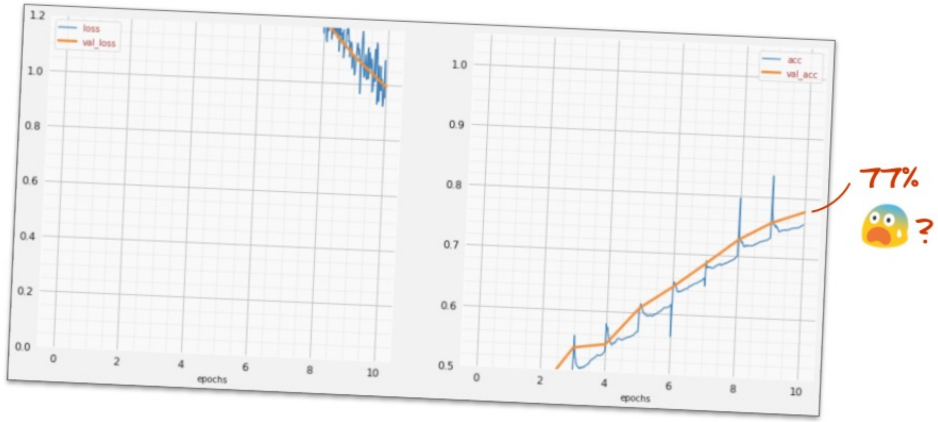
https://github.com/GoogleCloudPlatform/tensorflow-without-a-phd/blob/master/tensorflow-mnist-tutorial/keras_01_mnist.ipynb

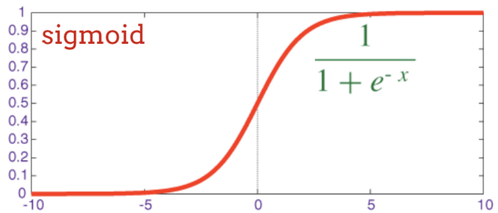
# Training Process



# Adding Layers



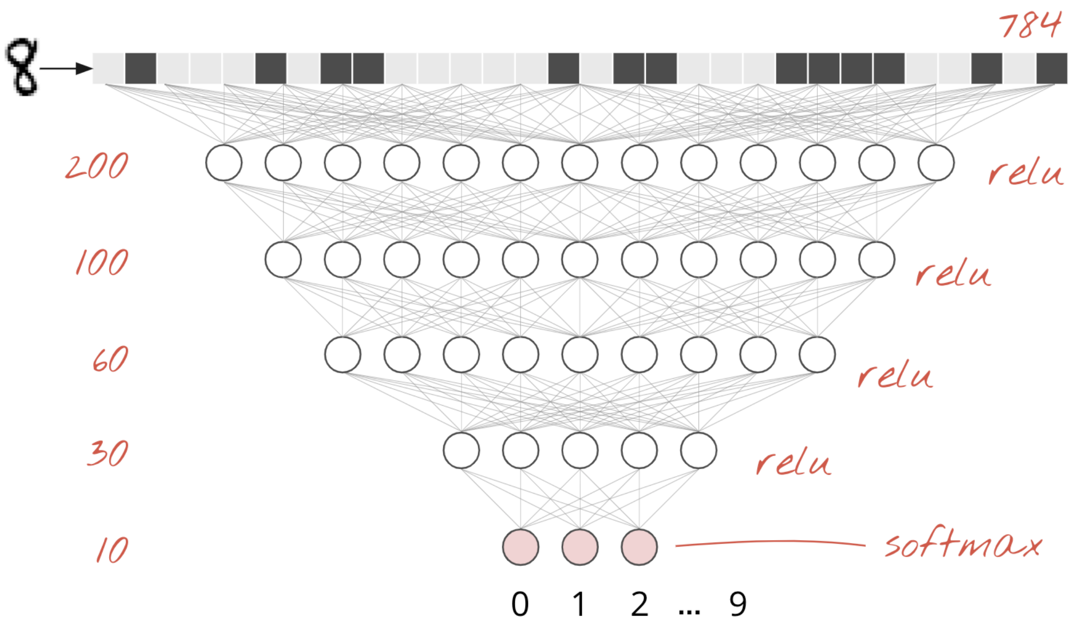




Getting flat

- The gradient can become very small and training get slower and slower.

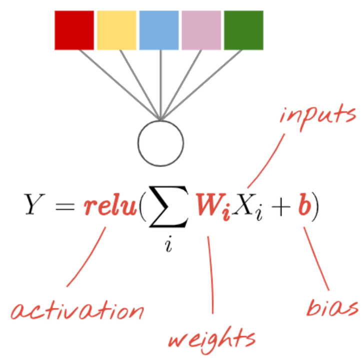
Simply adding more layers with sigmoid activations does not give us the expected results …

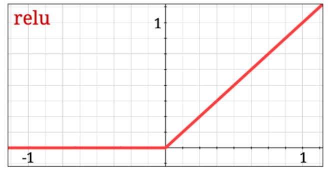
# Special Care for Deep Networks

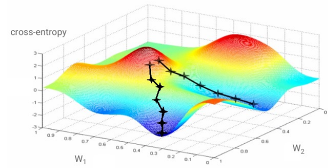



**HANDS ON**:

Replace all `activation='sigmoid'` with `activation='relu'` in your layers and train again.

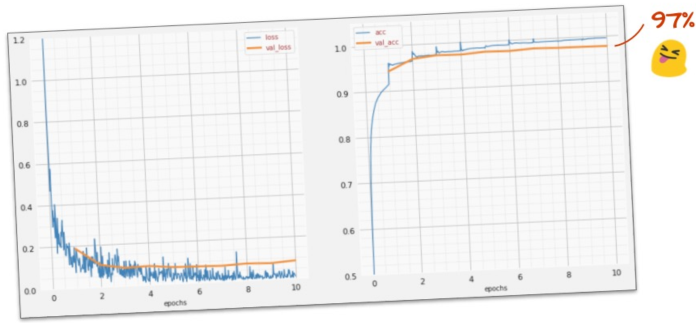



**HANDS ON**:

Replace the `'sgd'` optimizer with a better one, for example `'adam'` and train again.




https://github.com/GoogleCloudPlatform/tensorflow-without-a-phd/blob/master/tensorflow-mnist-tutorial/keras_02_mnist_dense.ipynb

# Lecture
# AI Meets Design II

Wan Fang
Southern University of Science and Technology

# Tool: Assessing feasibility for idea selection

Use this flowchart to quickly assess how feasible / viable your AI idea is

**Could you teach an intern how to do this task?**

- nah → If an intern can't do it, neither can a machine (for now). → But, it may be the type of task that only a machine can perform. Check in with an expert or quick Google search before giving up.
- sure ↓

**Do you own or are you able to gather (open source / public or buy) sets of the data you need?**

- no have → No data, no machine learning. Sorry. Go get yourself some data.
- got it ↓

**Is the data labelled and organized? Do you know what you're looking for?**

- no it's unsupervised → If you're out for detection, prediction, or generation, it's probably possible but might be costly $$. → (to: Do you have any in-house data scientists or machine learning engineers?)
- yes it's supervised ↓

**How many labelled examples do you have per category?**

- <5000 → Collect more data, brush yourself off and try again.
- >5000 ↓

**Do you have any in-house data scientists or machine learning engineers?**

- yass → What are you waiting for?!
- nope ↓

**Is there an open-source SDK of API for what you're trying to do?**

- nope
- I'm not sure / I can't find it but I've seen it in an app
- yass → It's looking real hopeful. Go forth.

# Activity:
# Framing your task
# for concept development

Google HCML team speak from experience when they say:
*"Find experts who can be the best possible teachers for your machine learner—people with domain expertise relevant to whatever predictions you're trying to make. We recommend that you actually hire a handful of them, or as a fallback, transform someone on your team into the role. We call these folks "content specialists" on our team."*

The strength of machine learning is that we don't have to program the rules explicitly. At this stage of the process, it is helpful to think about them and try construct a logic based on how we humans perform the task.

1. Start with the classic exercise: describe the way a human expert would perform the task or answer the question.

   If you were to ask 10 people, would they agree on the method (for the most part)? If some do it better or differently - what can we learn from their approach?

   Especially if what you're predicting is (highly) subjective, spend extra time on this step.

2. Imagine you're onboarding a new person for this job. What do they need to understand? What assumptions would you want them to make? How would you respond so they improve over time?

3. What's the nature of the task? Can you box it as an clustering, classification, or regression problem? Refer back to the crash course in the beginning of this toolkit to find the vocabulary. Knowing this will help you understand the task as well as communicate with your tech team.

In the example of Spotify's Discover Weekly, **the human expert** would be a music lover on the hunt for new music.

Do you have data of past well-executed and completed tasks? This could be used as an initial training data set.

**Tip**:
Draw a diagram of the current workflow including IFTT statements and data required to make decisions.

# Activity:
# Plotting your model for concept development

By plotting a simple flowchart, we can begin forming a rough idea of the inputs, outputs, and logic required for our model to create value. We're also surfacing our assumptions and unknowns in the process.

1

**Objective** - What is the question we're trying to answer and asking the machine?
**Output** - How is the machine's answer presented and interpreted?

2

**Features** - What data points do you need or are important factors in answering the question?
**Input** - Which data sets does that data reside in? What data will the model be trained on? What data does the user input?
**+** Draw connections between the assumed features and data sets they reside in.

3

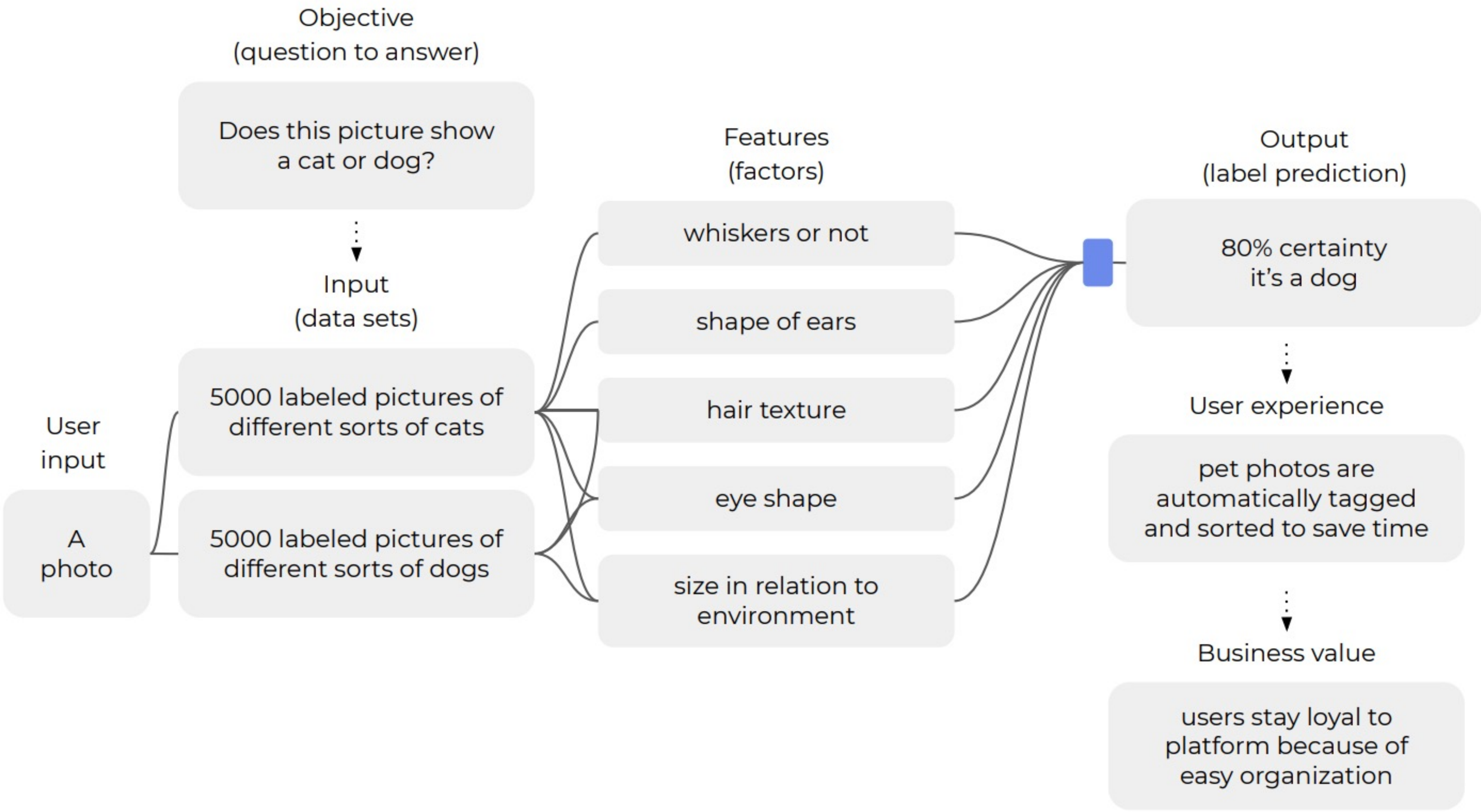
**User experience** - How does the outcome get presented to and help the user?
**Business value** - How does the solution return value to the organization?

AI answers (mostly) in probabilities with a confidence level. Formulate your output as a **probability**.

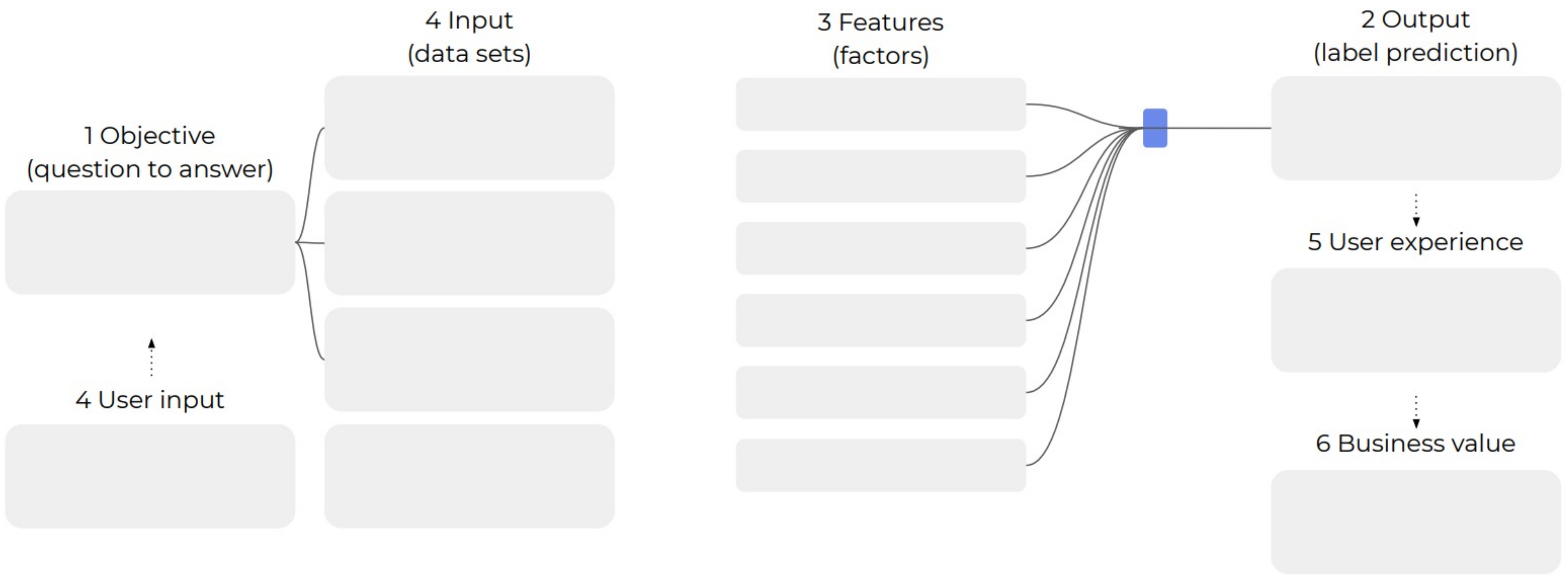
Do you know which features go into the answer? Think about the variables and patterns humans look at when performing this task or answering this question.

Do you have this **data to input**? If not, how do you acquire it?

# Activity:
# Plotting your model
# for concept development



# Worksheet:
# Plotting your model




**1 Objective**
What is the question we're trying to answer and asking the machine?

**2 Output**
How is the machine's answer presented and interpreted?

Formulate your output as a probability.

**3 Features**
What data points do you need or are important factors in answering the question?

Do you know which features go into the answer? Think about the variables and patterns humans look at when performing this task or answering this question.

**4 Input**
Which data sets does that data reside in? What data will the model be trained on? What data does the user input?

Do you have this data to input? If not, how do you acquire it?

**+ Connect**
Draw connections between the assumed features and data sets they reside in.

**5 User experience**
How does the outcome get presented to and help the user?

**6 Business value**
How does the solution return value to the organization?

# Prototyping + testing

You're with a handful of ideas and it's time to get more in-depth with your user research. Through prototyping and testing, you (in)validate your AI ideas and their design and implementation specs.

Do users want and need your solution? Are they open to adoption? Are they willing to share data and invest themselves into training the model (if necessary)? How can we test rather than just ask? How can we prototype the experience of adaptive intelligent systems?

In this chapter you will find:

**User research & feedback**
to know what to  inquire about in addition to the usual

**Prototyping & testing**
to explore how to prototype and test AI applications

# Activity:
# User research & feedback
# for assessing desirability

1

Assuming you did initial user research to inform your concepts so far, now it's time to go out and (in)validate your value proposition in more detail. First assess your need as you do for any problem, asking:

- What problem does it solve or opportunity does it tap into?
- Who benefits and in what scenario?
- How pressing is the problem? For how many?
- What do they gain from the new solution? How and how much better is it than the current solution? What other advantages do they see?

# Activity:
# User research & feedback for assessing desirability

Iterate on your value proposition statement based on your learnings and get ready to prototype for deeper insights.

2

Once you've validated that this is indeed a problem worth solving, gather insights about your users' perspective on the AI aspects of your concept(s).

**Mental models**
What are their notions about having an intelligent, adaptive system work for them? Are they willing to adopt it? How important is transparency? Depending on how visible your AI elements are, this might be more or less important.

**Defining success and failure**
How accurate must the model be to offer user value? How high are the costs of mistakes? What would best vs worst behavior look like?

**Machine teaching**
What does the user need to invest to get value out of the system? Are they willing to share the data your model needs? Are they willing to provide the necessary feedback and teach the model?

**Ethical & experiential concerns**
What concerns do they have? Do major ethical concerns arise? Unintended consequences, edge cases, and extreme users?

# Activity:
# Prototyping & testing
# for assessing desirability

1

**Prototype**

To test desirability, opt to simulate the experience without building the model and observing the responses.

Testing the concept offering can be done with product / service posters or app marketplace.

Common prototyping techniques for AI are:
Role playing
Wizard of Oz
Personalized wireframes.

Where possible, gather and use real-life personal data in your prototypes rather than placeholder content.

Provotypes (prototypes that provoke) can also be a great way to build an understanding of your users' needs.

*"Fake it till you make it. If forced to choose, it's leaps-and-bounds more useful to prototype your UX with a user's real content than it is to test with real ML models - as it affords you genuine insights into the way people will derive value and utility from your (theoretical) product."*

by Google Clips' team on UX of AI

# Activity:
# Prototyping & testing
# for assessing desirability

2

### Testing

Do user testing as usual and observe users' behavior. Ask them to think out loud as they're interacting with your artefact.

Keep in mind that while testing is important to understand your user, working with adaptive systems requires the designer to sacrifice a certain level of control over the final user experience exactly because it will adapt to each user and over time.

3

### Analysis & selection

Analyse and synthesize your findings. Based on all your findings, decide which idea(s) (if any) to move forward with.

It can help to revisit some of the activities in idea selection phase and reconsider feasibility, viability, desirability, and responsibility.

# Design
# + Implementation

You've developed the concept, validated with your users, and are ready to start building and bringing your idea into the world.

How to build it? How do you translate user needs to algorithmic parameters? What considerations do you need to make during design and implementation? How will you align needs across departments and stakeholders while keeping your user at the core?

In this chapter you will find:

**Defining success and failure ***
to understand confusion matrix and cost of falses

**UX and design challenges of AI ***
to learn about the unique design challenges of the material

**Mapping user needs to models**
to translate subjective user needs into model trade-offs

**Capturing design tensions ***
to explore the complexity of designing for human values

**Metrics to evaluate by ***
to agree on when your model will be good enough

**Consequence wheel ***
to anticipate (un)intended consequences and impact

To define what success looks like, it can help to imagine what failure looks like. Understanding cost of false, we can make conscious decisions about what to optimize for and what to trade-off.

# Activity:
# Defining success and accounting for failure

1

**Confusion matrix**

The confusion matrix (right) helps you map the impact of correct and false predictions.

True & false - The impact of misdiagnosing illness is much greater than the impact of misdiagnosing whether I like a song.

Positive & negative - In addition, the impact of diagnosing me with illness when I'm healthy, or failing to diagnose me when I am sick, is also different.

| | | Machine prediction | |
|---|---|---|---|
| | | Positive | Negative |
| User reality | Positive | :) True positive | :( False negative |
| | Negative | :( False positive | :) True negative |

Think through and discuss each of the 4 states and how they affect the your user and stakeholders. Re-clarify your objective. Which of the 4 states is most and least desired?

2

**Precision vs recall trade-off**

Building on that understanding, we can inform our position in common trade-off known as precision vs recall. While your engineers will work on both, at some point they'll need to make a decision on which to optimize for. In your use case, what is worse? Making a false prediction (optimize for precision), or missing true prediction (optimize for recall)?

**How important is ..**

**Precision**
% of predictions that are spot-on

VS

**Recall**
% of objects that are predicted

# Worksheet:
# Confusion matrix

**Machine prediction**

| **User reality** | Positive | Negative |
|---|---|---|
| Positive | :) What does a **true positive** look like? | :( What does a **false negative** look like? |
| Negative | :( What does a **false positive** look like? | :) What does a **true negative** look like? |

The confusion matrix is a diagram to help map the impact of correct and false predictions.

True & false
The impact of misdiagnosing illness is much greater than the impact of misdiagnosing whether I like a song.

Positive & negative
In addition, the impact of diagnosing me with illness when I'm healthy, or failing to diagnose me when I am sick, is also different.

Think through and discuss each of the 4 states and how they affect the your user and stakeholders. Re-clarify your objective. Which of the 4 states is most and least desired?

## How important is ..

**Accuracy**
% of predictions are correct

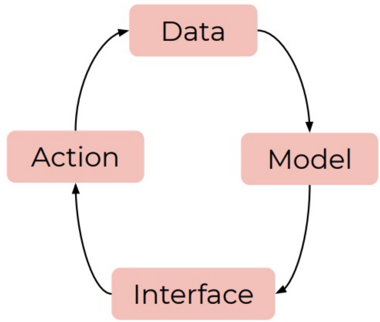
VS

**Transparency**
ability to trace back why/how

1 **Reward & penalize**
In the same way we train pets, you can reward or penalize the model. Think about what behavior to reward or penalize to direct your model towards ideal behavior. It can help to imagine you're onboarding a person for this task. How would you respond to them in any outcome so they improve for next time?

2 **Machine teaching**
Your user can play an active role in teaching the machine what is desired behavior through explicit and implicit feedback.

This way you can leverage what AI practitioners call "the virtuous cycle of data" (on the right) where more data means a better product, a better product means more users, which in turn means more data. Think about how you might set up these feedback loops in your interface.




3 **Accuracy vs transparency trade-off**
Most models suffer from a  trade-off between accuracy and transparency. How important is it for your user and your organization to understand why a certain prediction was made?

# Activity: Mapping user needs to models for machine teaching strategies

Optimizing for one thing always implies letting go of another. Once we have a clear objective in mind, we must train the model to understand and pursue such behavior.

By default, engineers will strive for maximum accuracy. In some cases, a lower score suffices at fulfilling a need, or the UX can make up for a range of errors. As designers, we can help define when models are or aren't 'good enough' to provide value to our user.

# Activity: Metrics to evaluate your model by

1

**Benchmarking**
Discuss and place on the spectrum between 1% and 99%:

**Current human benchmark** based on how people currently perform the task (optional: expert & layman)

**Baseline model** based on research and industry standard

**Minimum confidence level** based on cost of falses. How confident does the model need to be before presenting an answer? At what point is making the wrong prediction more harmful than not making one at all?

**Minimum benchmark to provide value to user** based on your use case and user research

**Targeted benchmark** based on the above and cost of falses. At what point is the prediction accurate enough to provide consistent value without risking user trust?

**100% accuracy**

**0% accuracy**

2

**Evaluating**
You can use these metrics to track the performance of your model on training and new data - and know when you're ready to deploy your first version for users.

# Worksheet: Benchmarking

**100% accuracy**

Discuss and place on the spectrum between 1% and 99%:

**Current human benchmark** based on how people currently perform the task (optional: expert & layman)

**Baseline model** based on research and industry standard

**Minimum confidence level** based on cost of falses. How confident does the model need to be before presenting an answer? At what point is making the wrong prediction more harmful than not making one at all?

**Minimum benchmark to provide value to user** based on your use case and user research

**Targeted benchmark** based on the above and cost of falses.

At what point is the prediction correct enough to provide value? How confident does the model need to present an answer? At what point is making the wrong prediction more harmful than not making one at all?

**0% accuracy**

Below, we identify and introduce you to 9 challenges in designing intelligent, adaptive, and (semi-)autonomous systems:

Each design material comes with unique challenges. In the same way that designing a poster is different from designing a mobile app, designing AI-driven applications is different from designing apps.

### User Trust & Transparency

1. **Explainability**
Making sense of the machine and communicating to the user why the system acts the way it does

2. **Managing expectations**
Helping the user understand what the system can and can not do (over time) by being transparent about abilities and limitations and building helpful mental models of it

3. **Graceful failure & accountability**
Assume failure and design graceful recoveries. Take accountability for mistakes and minimize cost of falses for your user

### User Autonomy & Controls

4. **Machine teaching & user feedback**
Allowing the user to teach the machine with implicit and explicit feedback loops and collecting direct data input

5. **User controls & customization**
Giving users the controls to customize the system/algorithm to their needs and intervene with the course of a model if needed

6. **Data privacy & security**
Collect, handle, and store user data with care. Be transparent about who can access what data and why while acknowledging their ownership

### Value Alignment

7. **Computational virtue**
Translating subjective human needs, values, and experiences into algorithmic parameters the model can optimize for

8. **Bias & inclusivity**
Mitigating bias and guarding inclusivity in data and models to ensure fair treatment for all

9. **Ethics & (un)intended consequences**
Unprecedented scale, speed and complexity call for a new level of thoughtfulness and responsibility in anticipating impact and (un)intended consequences

# Resource: UX and design challenges of AI as a design material

# Worksheet: UX of AI challenges

| | | | |
|---|---|---|---|
| 1. Explainability - How will we help our user understand certain outcomes? | 2. Managing expectations - How will we establish realistic expectations? | 3. Graceful failure & accountability - How will we design for trust in case of failure? | **User Trust & Transparency** |
| 4. User feedback - How will your user provide feedback to the system? | 5. User autonomy - How will the user be able to customize their experience? | 6. Data privacy & security - How will you collect, store, and handle data? | **User Autonomy & Control** |
| 7. Computational translation - How will you turn needs into parameters? | 8. Bias & inclusivity - How will you prevent bias and guard inclusivity? | 9. Ethics & (un)intended consequences - How will you look out for negative and positive impact? | **Value Alignment** |
| 10. Which other (design) challenges do you foresee? | | | |

# Tool: Capturing design tensions and value polarities

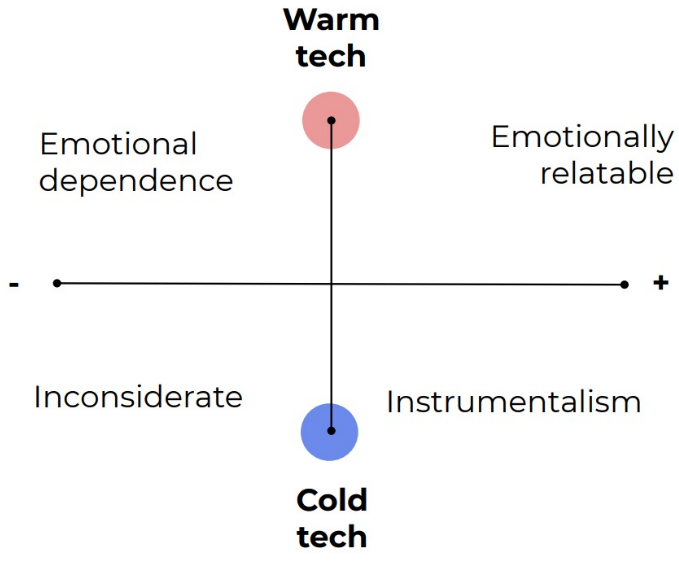
Design ethics, as general ethics, aren't as simple as wrong and right. More often, we are designing for a tension between different values.

Making these polarities explicit can help us design for them in a constructive way: deciding where on the spectrum we want our product or service to reside, on what to foster, and look out for.

1. Name and write the polarizing values on the top and bottom. Iterate on the terms as you go to capture their essence.

2. For both values, think about all the positives and negatives. Write them into the boxes or on post-its first. Questions to get started:

   What opportunities and benefits are present? What challenges and threats? Why would users want this? Why not? What human values are at play? Human rights even?




how might we relate to tech emotionally in a healthy way?

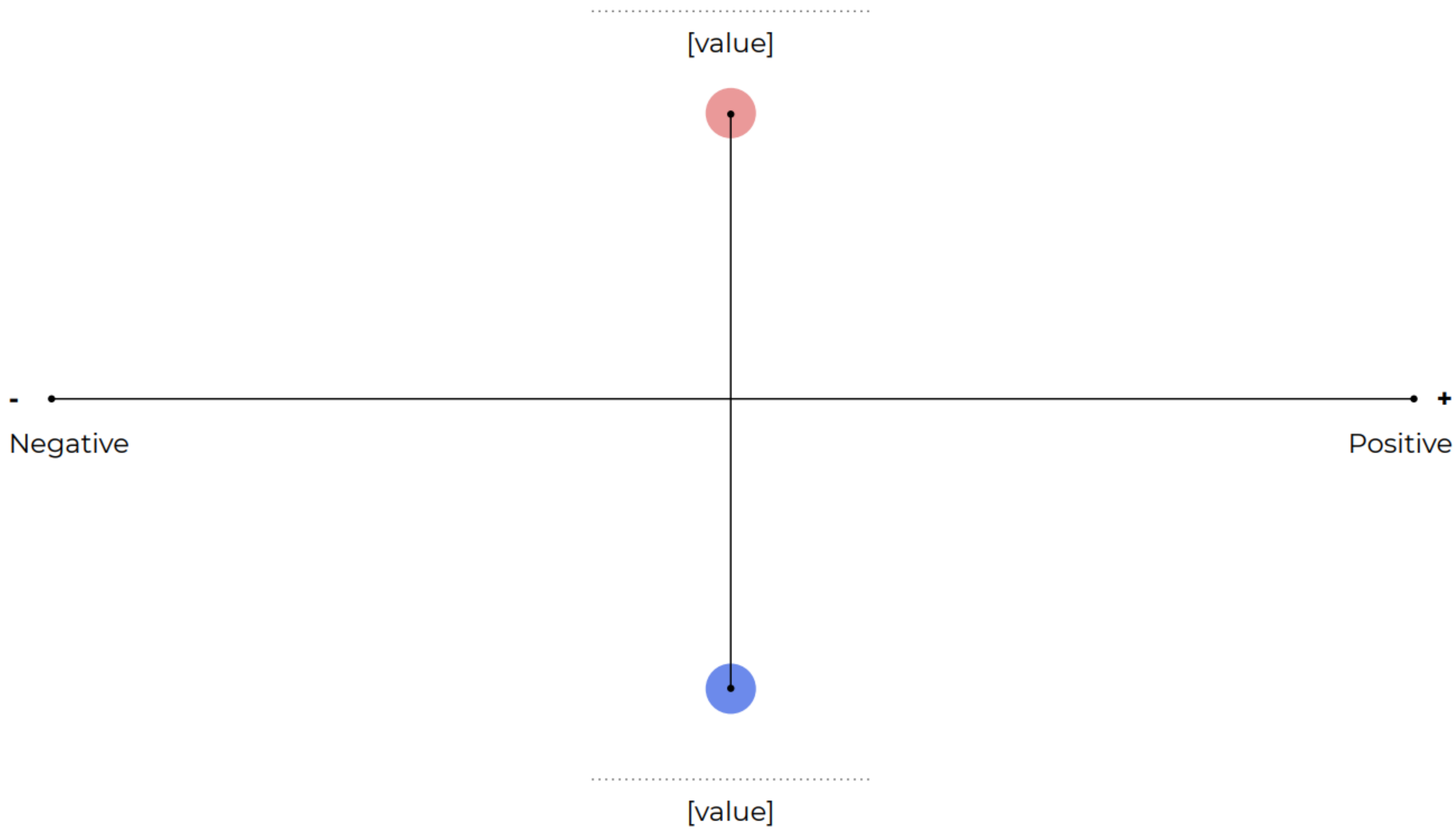
3. Give each of the dimensions a term to summarize its sentiment.

4. Considering positive and negative aspects, draw an x where you think your product or service should reside. If you're working in a team, let everyone do this individually and then discuss and agree on differences.

   Repeat making polarity maps as many times as needed and allowed depending on your context. Any product would have at least 3-5 polarities.

5. Think about how to harness the positive and how to limit the negative aspects. Keep this mental model front and centre as you continue to design, develop, and deploy your AI-driven application.

# Worksheet:
# Value polarities




[design question]

Name and write the polarizing values on the top and bottom. Iterate on the terms as you go to capture their essence.

For both values, think about all the positives and negatives. Write them into the boxes or on post-its first.

Questions to get started: What opportunities and benefits are present? What challenges and threats? Why would users want this? Why not? What human values are at play? Human rights even?

Give each of the dimensions a term to summarize its sentiment.

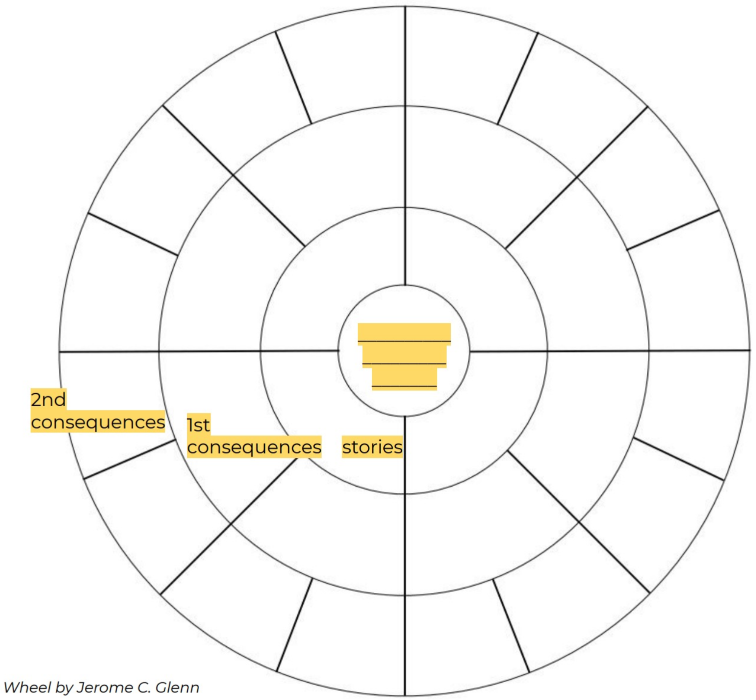
Considering positive and negative aspects, draw an x where you think your product or service should reside. If you're working in a team, let everyone do this individually and then discuss and agree on differences.

Repeat making polarity maps as many times as needed and allowed depending on your context. Any product would have at least 3-5 polarities.

Think about how to harness the positive and how to limit the negative aspects. Keep this mental model front and centre as you continue to design, develop, and deploy your AI-driven application.

## Guiding questions:

- What consequences & impact might this have? Think of culture, politics, economics, etc.
- What's the best-case scenario? Worst case? Who benefits? Who suffers?
- Is this a world you'd like to live in? Why or why not?
- Which industry, services, or social rituals might be disrupted by this?
- Which new user pain points or opportunities exist in this world?




*Futures Wheel by Jerome C. Glenn*

1. Start with your product, service, interaction, or AI-driven application in the middle.
2. Write a handful of user stories and experiences in the second row.
3. Begin thinking through consequences by asking "if this then what?" as you work outward. Consider different groups, scenarios, and lenses. Do this for at least 3 and ideally more levels.

# Tool: Consequence wheel to anticipate (unintended) consequences

# Worksheet: Consequence wheel

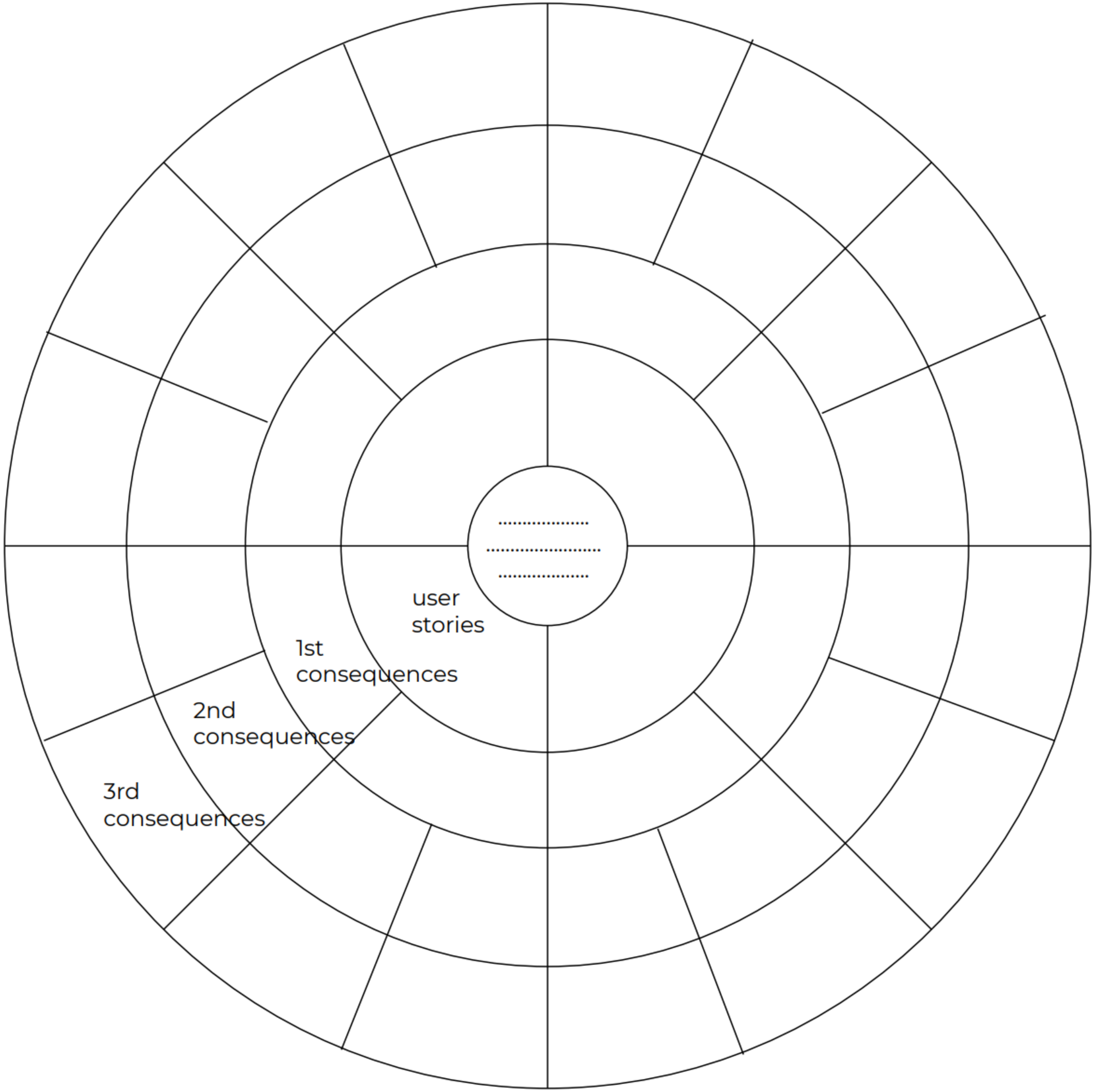
Start with your product, service, interaction, or AI-driven application in the middle.

Write a handful of user stories and experiences in the second row.

Begin thinking through consequences by asking "if this then what?" as you work outward. Consider different groups, scenarios, and lenses. Do this for at least 3 and ideally more levels.

**Guiding questions:**

- What consequences and impact might this have? Think of culture, politics, economics, ecology, etc.
- What's the best case scenario? Worst case? Who benefits? Who suffers?
- Is this a world you'd like to live in? Why or why not?
- Which industry, services, or social rituals might be disrupted by this?
- Which new user pain points or opportunities exist in this world?



## To learn

If you want learn more about AI, here are some of my favorite learning resources:

- Elementsofai.com
- People + AI Guidebook by Google
- AI for Everyone on Coursera with Andrew Ng
- AI-driven design e-book series by Adyen & AWWWARDS
- Algorithms.design
- Machine Learning for Designers by Patrick Hebron for O'REILLY

## To apply

If you want to explore more design tools, check out these projects:

- Intelligence Augmentation Design
- Toolkit by Futurice
- AI Ethics cards by IDEO
- Machine Ethics Toolkit
- IBM AI Camp DIY Guide

## To play

If you're feeling a little overwhelmed, here are a few of my favorite AI games and experiments you can play with:

- Teachable Machine by Google
- Emoji Scavenger Hunt by Google
- Quick, Draw! by Google
- Runway ML

# Additional resources

If this toolkit has succeeded at peeking your interest and you're hungry for more, these are additional resources to continue your learning journey.

**To build**

Plenty MLaaS (Machine Learning as a Service), and cloud platform solutions have become available to cater to different goals and levels of technical expertise.

| Type of dev | Pre-trained models on their data | Train models on own data | GUI for custom models | Libraries for custom models |
|---|---|---|---|---|
| Tools | APIs & SDKs. Mostly by Amazon, Microsoft, Google, IBM. Check rapidapi.com | AutoML by Google, Microsoft Azure ML Studio, Amazon ML | Amazon Sagemaker, IBM Watson ML Studio | Tensorflow, sci-kit, pytorch, Keras |

# Additional resources

If this toolkit has succeeded at peeking your interest and you're hungry for more, these are additional resources to continue your learning journey.

# Day 02
# AI Meets Design II

**Thank you~**

Wan Fang
Southern University of Science and Technology