Image Classification

Object Recognition

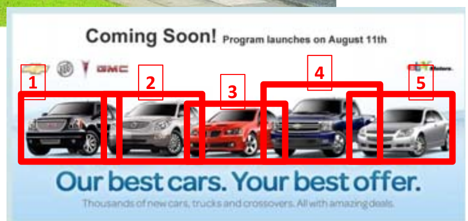
- Is that a car?
  - Verification
- What is it?
  - Instance
  - Category
- Where is it?
  - Localization
  - Segmentation
- How many are there?

Instance: **Car**

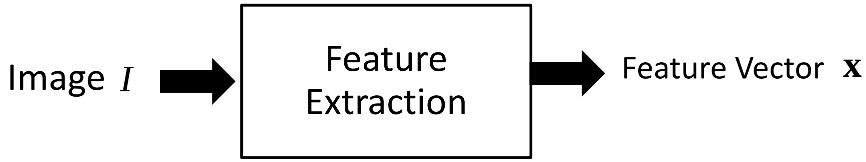
Category: **Vehicle**

Object Recognition

- Is that a car?
  - Verification
- What is it?
  - Instance
  - Category
- Where is it?
  - Localization
  - Segmentation
- How many are there?



Feature Space



- Given an image $I(x, y)$
- Transform it into 1-D vector called features or descriptor

Feature Space

1	1	1	1	1	1	1	1
1	1	1	0	0	1	1	1
1	1	0	0	0	1	1	1
1	1	0	1	0	0	1	1
1	0	0	1	1	0	1	1
1	0	0	1	1	0	0	1
0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
1	1	1	1	1	1	1	1

Image I_1

Feature Vector

$\mathbf{x}_1 = [8\ \ 7\ \ 12\ \ 13]$

Count the number of zero in 4 blocks

Feature Space

1	1	1	1	1	1	1	1
1	1	0	0	0	0	1	1
1	0	0	0	0	0	0	1
1	0	0	0	1	1	1	1
1	0	0	1	1	1	1	1
1	0	0	1	1	1	1	1
1	0	0	0	0	1	1	1
1	0	0	0	0	0	1	1
1	0	0	1	1	1	1	1
1	0	0	1	1	1	1	1
1	0	0	1	1	1	1	1
1	0	1	1	1	1	1	1

Image I_2

Feature Vector

$\mathbf{x}_2 = [12\ \ 5\ \ 13\ \ 3]$

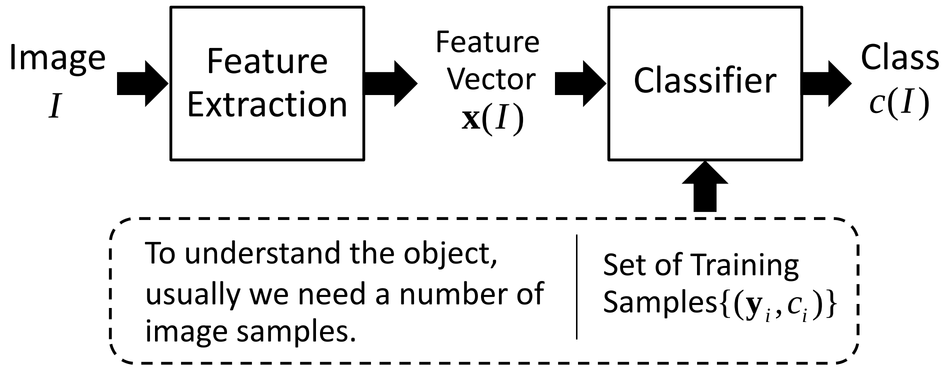
Count the number of zero in 4 blocks

Feature Extraction Method

- Color Histogram
- HOG
- GLCM
- LBP
- GIST
- Hu's moments
- Projection Profile
- Zoning
- Fourier Descriptor
- Stroke Width
- Wavelet Transform
- Haar-like feature
- Bag-of-Words

Classification

Create decision function to determine class of images from their feature



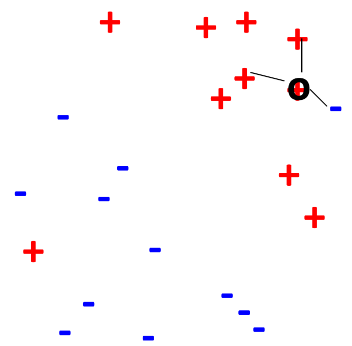
$\mathbf{y}_i$ = Feature vector extracted from the i – th sample image c_i = Class of the i – th sample image

Classification Methods

- K-Nearest Neighbors (KNN)
- Decision Tree
- Naïve Bayes Classifier
- Support Vector Machine (SVM)
- Artificial Neural Network (ANN)
- AdaBoost

KNN

k-nearest neighbors



- Memorize all training data
- Find K closest points to the query
- The neighbors vote for the label

For example (K=3):
Vote(+)=2 Vote(−)=1

KNN

Unknown Class Image → [0.21 0.29 0.28 0.21]

Find distance (e.g. Euclidean)

Training Samples

X	[0.30 0.21 0.17 0.32]	0.20
X	[0.23 0.26 0.24 0.27]	0.08
X	[0.25 0.25 0.24 0.26]	0.09
y	[0.38 0.28 0.23 0.11]	0.20
y	[0.32 0.25 0.25 0.18]	0.12
y	[0.29 0.30 0.28 0.14]	0.11
Z	[0.14 0.33 0.33 0.19]	0.10
Z	[0.15 0.34 0.34 0.18]	0.10
Z	[0.20 0.29 0.28 0.21]	0.01

$K = 1$ → Class Z

KNN

Unknown Class Image → [0.21 0.29 0.28 0.21]

Find distance (e.g. Euclidean)

Training Samples

X	[0.30 0.21 0.17 0.32]	0.20
X	[0.23 0.26 0.24 0.27]	0.08
X	[0.25 0.25 0.24 0.26]	0.09
y	[0.38 0.28 0.23 0.11]	0.20
y	[0.32 0.25 0.25 0.18]	0.12
y	[0.29 0.30 0.28 0.14]	0.11
Z	[0.14 0.33 0.33 0.19]	0.10
Z	[0.15 0.34 0.34 0.18]	0.10
Z	[0.20 0.29 0.28 0.21]	0.01

$K = 3$ → Class X

KNN

Unknown Class Image → $[0.21\ 0.29\ 0.28\ 0.21]$

↓ Find distance (e.g. Euclidean)

Training Samples

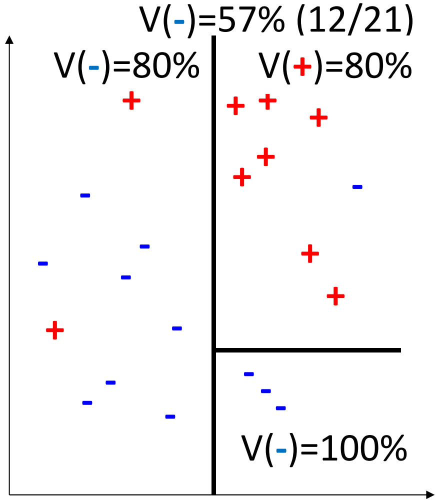
x	$[0.30\ 0.21\ 0.17\ 0.32]$	0.20
x	$[0.23\ 0.26\ 0.24\ 0.27]$	0.08
x	$[0.25\ 0.25\ 0.24\ 0.26]$	0.09
y	$[0.38\ 0.28\ 0.23\ 0.11]$	0.20
y	$[0.32\ 0.25\ 0.25\ 0.18]$	0.12
y	$[0.29\ 0.30\ 0.28\ 0.14]$	0.11
z	$[0.14\ 0.33\ 0.33\ 0.19]$	0.10
z	$[0.15\ 0.34\ 0.34\ 0.18]$	0.10
z	$[0.20\ 0.29\ 0.28\ 0.21]$	0.01

$K = 5$

↓

Class Z

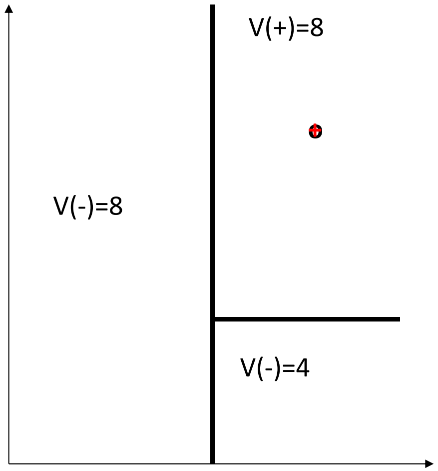
Decision Tree

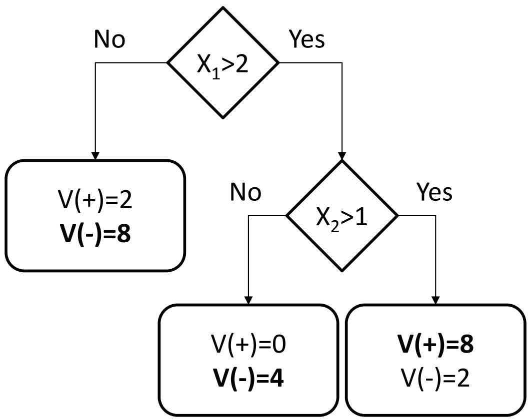


- Partition data into pure chunks
- Choose rules to make **purity** of child nodes as high as possible.
- Split the training data
  - Build left tree
  - Build right tree
- Count the examples in the leaves to get the votes: V(+), V(-)
- Stop when
  - Purity is high
  - Data size is small
  - At fixed level

261458 & 261753 Computer Vision #6

Decision Tree

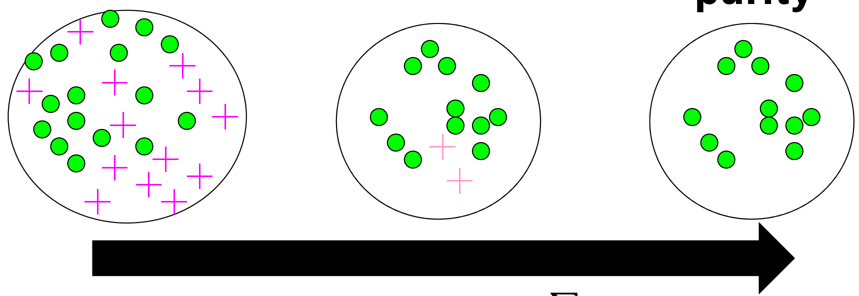




261458 & 261753 Computer Vision #6

Purity Measure

Very impure group | **More purity** | **Maximum purity**



Reduction in Entropy $-\sum_i p_i \log_2 p_i$

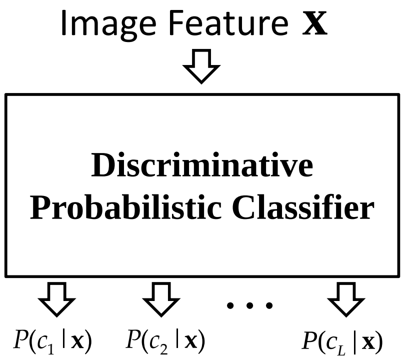
Increment in Gini Coefficient $1-\sum_i (p_i)^2$

http://www.cs.washington.edu/education/courses/cse415/06wi/notes/DecisionTrees.ppt

261458 & 261753 Computer Vision #6

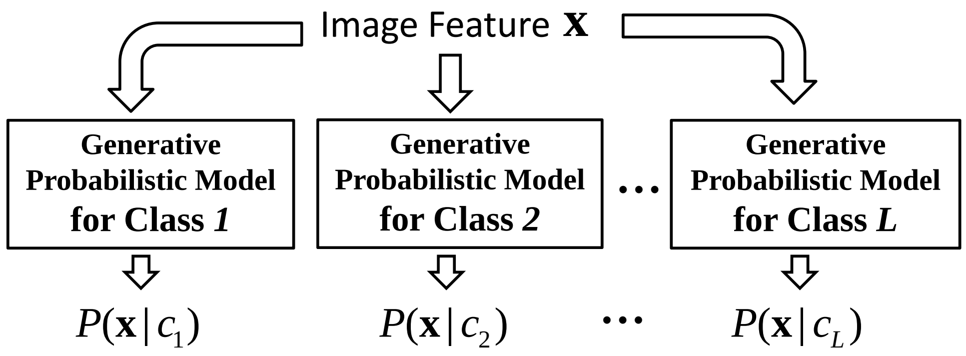
Probabilistic Classification

- Establishing a probabilistic model for classification
  - **Discriminative model** $P(C = c \mid X = \mathbf{x})$



Probabilistic Classification

- **Generative model** $P(X = \mathbf{x} \mid C = c)$



Probabilistic Classification

- Prior, conditional and joint probability for random variables
  - Prior probability: $P(C = c)$
  - Posterior probability $P(C = c \mid X = \mathbf{x})$
  - Likelihood: $P(X = \mathbf{x} \mid C = c)$
  - Joint probability: $P(X = \mathbf{x}, C = c)$
  - Relationship: $P(X, C) = P(X \mid C)P(C) = P(C \mid X)P(X)$

- Bayesian Rule

$$P(C = c \mid X = \mathbf{x}) = \frac{P(X = \mathbf{x} \mid C = c)P(C = c)}{P(X = \mathbf{x})} \qquad Posterior = \frac{Likelihood \times Prior}{Evidence}$$

Probabilistic Classification

- MAP classification rule
  - **MAP**: **M**aximum **A** **P**osterior
  - Assign ***x*** to c^* if

$$P(C = c^* \mid X = \mathbf{x}) > P(C = c \mid X = \mathbf{x}) \quad c \neq c^*, \ c = c_1, \cdots, c_L$$

- Generative classification with the MAP rule
  - Apply Bayesian rule to convert them into posterior probabilities

$$P(C = c_i \mid X = \mathbf{x}) = \frac{P(X = \mathbf{x} \mid C = c_i)P(C = c_i)}{P(X = \mathbf{x})}$$
$$\propto P(X = \mathbf{x} \mid C = c_i)P(C = c_i)$$
$$\text{for } i = 1, 2, \cdots, L$$

  - Then apply the MAP rule

Naive Bayes Classifier

- Bayes classification for $\mathbf{x}=[x_1, x_2, \cdots, x_n]$

$$P(C \mid X=\mathbf{x}) \propto P(X=\mathbf{x} \mid C)P(C) = P(x_1, \cdots, x_n \mid C)P(C)$$

Difficulty: learning the joint probability $P(x_1, \cdots, x_n \mid C)$

- Naïve Bayes classification
  - Assumption that all input attributes are independent!

$$P(x_1, x_2, \cdots, x_n \mid C) = P(x_1 \mid C)P(x_2 \mid C) \cdots P(x_n \mid C)$$

  - MAP classification rule

$$[P(x_1 \mid c^*) \cdots P(x_n \mid c^*)]P(c^*) > [P(x_1 \mid c) \cdots P(x_n \mid c)]P(c), \quad c \neq c^*, c = c_1, \cdots, c_L$$

Naive Bayes Classifier

- Example: Play Tennis

Outlook=*Sunny*, Temperature=*Cool*, Humidity=*High*, Wind=*Strong*

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Naive Bayes Classifier

- Learning Phase

Outlook	Play=*Yes*	Play=*No*
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Temperature	Play=*Yes*	Play=*No*
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Humidity	Play=*Yes*	Play=No
High	3/9	4/5
Normal	6/9	1/5

Wind	Play=*Yes*	Play=*No*
Strong	3/9	3/5
Weak	6/9	2/5

P(Play=*Yes*) = 9/14 P(Play=*No*) = 5/14

Naive Bayes Classifier

- Test Phase
  - Given a new instance,

x'=(Outlook=*Sunny*, Temperature=*Cool*, Humidity=*High*, Wind=*Strong*)

  - Look up tables

P(Outlook=*Sunny*|Play=*Yes*) = 2/9
P(Temperature=*Cool*|Play=*Yes*) = 3/9
P(Huminity=*High*|Play=*Yes*) = 3/9
P(Wind=*Strong*|Play=*Yes*) = 3/9
P(Play=*Yes*) = 9/14

P(Outlook=*Sunny*|Play=*No*) = 3/5
P(Temperature=*Cool*|Play==*No*) = 1/5
P(Huminity=*High*|Play=*No*) = 4/5
P(Wind=*Strong*|Play=*No*) = 3/5
P(Play=*No*) = 5/14

  - MAP rule

P(*Yes*|**x**'): [P(*Sunny*|*Yes*)P(*Cool*|*Yes*)P(*High*|*Yes*)P(*Strong*|*Yes*)]P(Play=*Yes*) = 0.0053

P(*No*|**x**'): [P(*Sunny*|No) P(*Cool*|No)P(*High*|*No*)P(*Strong*|*No*)]P(Play=*No*) = 0.0206

Given the fact P(*Yes*|**x**') < P(*No*|**x**'), we label **x**' to be "*No*".

Naive Bayes Classifier

- Continuous-valued Input Attributes
  - Numberless values for an attribute
  - Conditional probability modeled with the normal distribution

$$\hat{P}(x_j \mid C = c_i) = \frac{1}{\sqrt{2\pi}\sigma_{ji}} \exp\left(-\frac{(x_j - \mu_{ji})^2}{2\sigma_{ji}^2} \right)$$

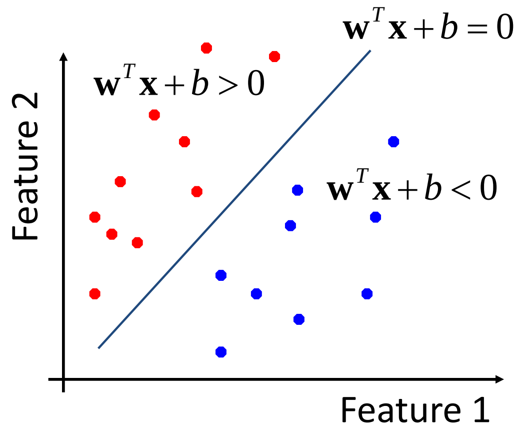
μ_{ji} : mean (avearage) of attribute values x_j of examples for which $C = c_i$

σ_{ji} : standard deviation of attribute values x_j of examples for which $C = c_i$

  - Learning Phase:
    - Use training sample to estimate μ_{ij}, σ_{ij}
  - Test Phase:
    - Calculate conditional probabilities with all the normal distributions
    - Apply the MAP rule to make a decision

SVM

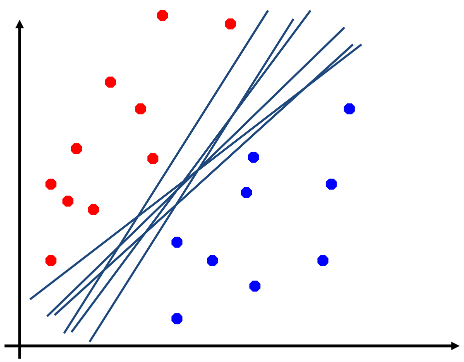
Support Vector Machine



- Typical SVM is binary classifier
- Decision model is described by optimal separating hyper-plane

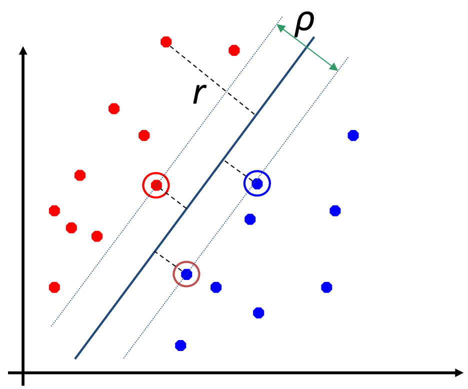
$$f(\mathbf{x}) = sign(\mathbf{w}^T\mathbf{x} + b)$$

SVM



Which of the linear separators is optimal?

SVM



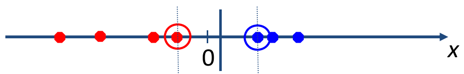
- Distance from example $\mathbf{x}_i$ to the separator is

$$r = \frac{\mathbf{w}^T\mathbf{x}_i + b}{\|\mathbf{w}\|}$$

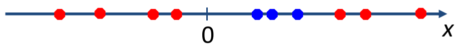
- Examples closest to the hyperplane are ***support vectors***.
- ***Margin*** ρ of the separator is the distance between support vectors.
- Find hyper-plane which maximizes the margin

Non-linear SVM

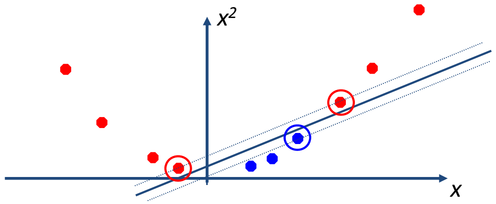
- Datasets that are linearly separable with some noise work out great:



- But what are we going to do if the dataset is just too hard?

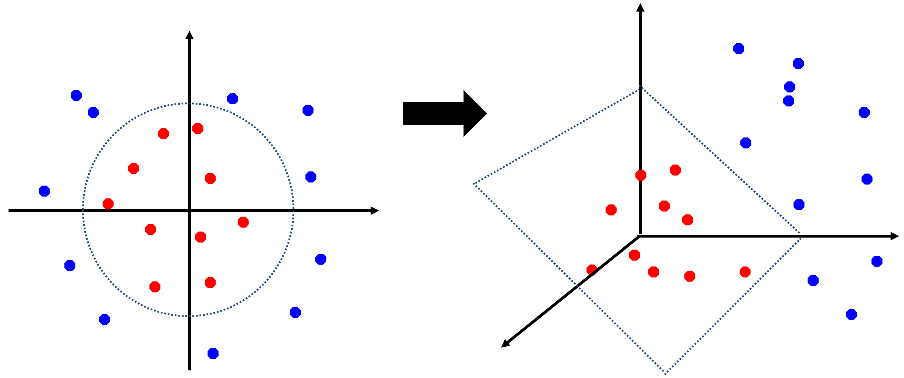


- How about… mapping data to a higher-dimensional space:

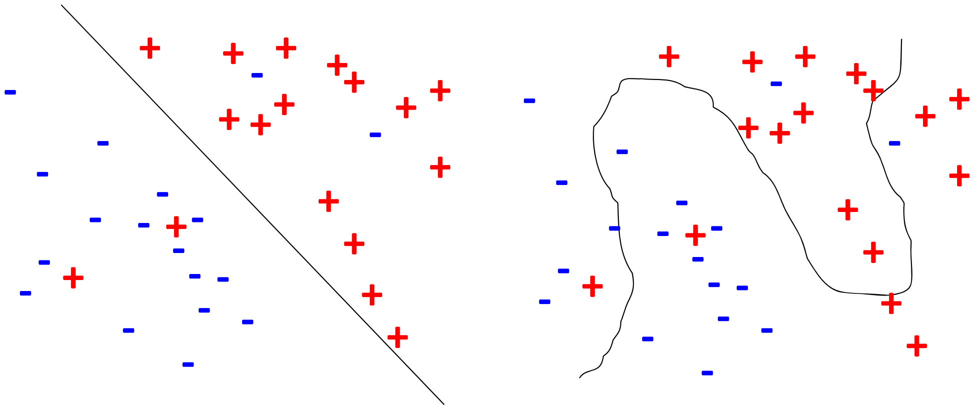


Non-linear SVM

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



Overfitting

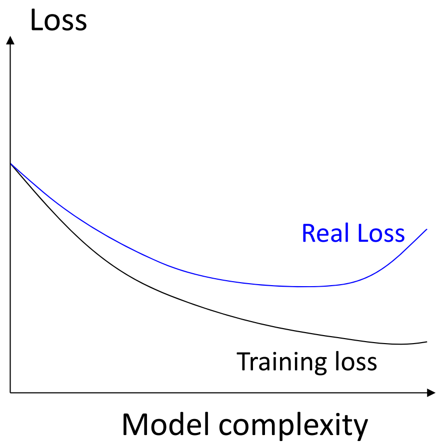


- Let's get more data
- Simple model has better generalization

Overfitting

- As complexity increases, the model *overfits* the data
- Training loss decreases
- Real loss increases
- We need to penalize model complexity

  = to regularize



Overfitting

- Split the dataset
  - Training set
  - Validation set
  - Test set
- Use training set to **optimize** model parameters
- Use validation test to **choose** the best model
- Use test set only to **measure** the expected loss