

CSC321 Neural Networks and Machine Learning

Lecture 3

January 22, 2020

Agenda

First hour:

- ▶ Multi-class classification
- ▶ Feature Mapping

Second hour:

- ▶ k-Nearest Neighbours
- ▶ Generalization

Announcement

- ▶ Homework 2 is due tomorrow
- ▶ Project 1 is due next week
- ▶ Homework 1 is graded
 - ▶ Very well done!
 - ▶ Solutions are on **Quercus**
 - ▶ Remark request due Jan 28th, 9pm on **Markus**

Review

1. In a supervised learning setup, what does $x_j^{(i)}$ represent?
2. What is the shape of the vector $\frac{\partial \mathcal{E}}{\partial \mathbf{w}}$?
3. What does α represent in the gradient descent step:
$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial \mathcal{E}}{\partial \mathbf{w}}$$
4. What can happen if α is too large? Too small?
5. What is the **batch size**? What happens if it is too large? Too small?

Classification

Classification Setup

- ▶ Data: $(x^{(1)}, t^{(1)}), (x^{(2)}, t^{(2)}), \dots (x^{(N)}, t^{(N)})$
- ▶ The $x^{(i)}$ are called *inputs*
- ▶ The $t^{(i)}$ are called *targets*

In classification, the $t^{(i)}$ are discrete.

In binary classification, we used the labels $t \in \{0, 1\}$. Training examples with

- ▶ $t = 1$ is called a **positive example**
- ▶ $t = 0$ is called a **negative example** (sorry)

Multi-class classification

Instead of there being two targets (pass/fail, cancer/not cancer, before/after 2000), we have $K > 2$ targets.

Example:

- ▶ Beatles ($K = 4$):
 - ▶ John Lennon, Paul McCartney, George Harrison, Ringo Starr
- ▶ Pets ($K = \text{something large}$):
 - ▶ cat, dog, hamster, parrot, python, ...

Representing the targets

We use a **one-hot vector** to represent the target:

$$\mathbf{t} = (0, 0, \dots, 1, \dots, 0)$$

This vector contains $K - 1$ zeros, and a single 1 somewhere.

Each *index* (column) in the vector represents one of the classes.

Representing the prediction

The prediction \mathbf{y} will also be a vector. Like in logistic regression there will be a linear part, and an activation function.

Linear part: $\mathbf{z} = \mathbf{W}^T \mathbf{x} + \mathbf{b}$

So far, this is like having K separate logistic regression models, one for each element of the one-hot vector.

Q: What are the shapes of \mathbf{z} , \mathbf{W} , \mathbf{x} and \mathbf{b} ?

Activation Function

Instead of using a *sigmoid* function, we instead use a **softmax activation** function:

$$y_k = \text{softmax}(z_1, \dots, z_K)_k = \frac{e^{z_k}}{\sum_{m=1}^K e^{z_m}}$$

The predictions y_k is now a **probability distribution** over the classes!

Why softmax?

- ▶ Softmax is like the multi-class equivalent of sigmoid
- ▶ Softmax is a continuous analog of the “argmax” function
- ▶ If one of the z_k is much larger than the other, then the softmax will be approximately the argmax, in the one-hot encoding

Cross-Entropy Loss

The cross-entropy loss naturally generalizes to the multi-class case:

$$\begin{aligned}\mathcal{L}(\mathbf{y}, \mathbf{t}) &= - \sum_{k=1}^K t_k \log(y_k) \\ &= -\mathbf{t}^T \log(\mathbf{y})\end{aligned}$$

Recall that only one of the t_k is going to be 1, and the rest are 0.

Summary

Hypothesis

$$\mathbf{y} = \text{softmax}(\mathbf{W}^T \mathbf{x} + \mathbf{b})$$

Loss

Function

$$\mathcal{L}(\mathbf{y}, \mathbf{t}) = -\mathbf{t}^T \log(\mathbf{y})$$

Optimization

Problem

$$\min_{\mathbf{W}, \mathbf{b}} \mathcal{E}(\mathbf{W}, \mathbf{b})$$

Gradient

Descent

$$\mathbf{W} \leftarrow \mathbf{W} - \alpha \frac{\partial \mathcal{E}}{\partial \mathbf{W}}, \mathbf{b} \leftarrow \mathbf{b} - \alpha \frac{\partial \mathcal{E}}{\partial \mathbf{b}}$$

Example: Beatle Recognition

Given a 100x100 pixel colour image of a face of a Beatle, identify the Beatle



Four possible labels:

- ▶ John Lennon
- ▶ Paul McCartney
- ▶ George Harrison
- ▶ Ringo Starr

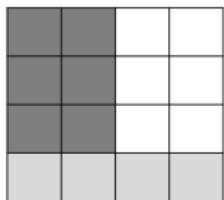
Aside: Representing an image

This is what John Lennon looks like to a computer:

26	128	127	124	123	125	126	141	215	217	137	82	69	33	34	49	28	19	32	30	28	29	40	39	31	24	33	33	43	36	63	58	71	54	68	77
132	113	151	172	184	194	183	175	150	147	142	90	96	100	101	100	98	98	103	107	104	181	185	150	90	79	61	66	46	29	25	17	9			
76	91	93	95	88	79	71	61	66	59	59	86	91	108	78	174	164	156	164	181	190	202	194	163	155	151	149	33	38	41	44	46	46	46		
29	33	59	54	44	41	65	64	72	77	68	80	83	72	87	93	101	106	95	85	83	72	71	63	61	59	42	17	165	189	174	174	172	188		
28	26	23	35	96	126	98	92	70	63	57	50	35	21	22	65	61	76	109	102	102	105	93	110	92	87	89	90	97	110	116	104	96	98	107	
10	206	195	172	156	156	148	145	27	28	28	31	32	32	35	26	33	44	80	80	77	62	39	28	40	34	51	69	44	78	94	98	87	64	54	
75	54	42	38	56	69	74	127	175	182	188	184	183	194	202	182	165	160	153	146	142	33	40	47	52	58	63	65	67	65	82	78	78			
63	73	105	122	102	122	136	145	147	97	52	35	23	11	13	36	62	66	80	109	126	132	178	167	186	180	187	190	194	188	164	158	152			
26	153	172	141	80	82	70	75	74	59	61	42	24	16	15	52	21	62	88	106	123	128	153	151	118	114	150	127	70	63	29	11	36	32		
90	189	170	150	148	159	158	153	150	135	144	148	149	151	151	151	150	145	158	203	179	85	96	52	88	67	57	51	61	56	58	37	14	55		
48	20	61	72	53	54	42	29	57	59	99	100	96	122	119	154	174	178	189	174	159	152	144	149	151	154	153	122	125	123	118	119	12			
45	47	48	52	65	64	82	69	75	68	74	85	110	112	111	127	139	117	73	53	55	84	68	60	61	50	52	51	52	80	82	84	127	141	45	
1	123	125	123	121	120	123	155	199	159	104	18	57	48	40	48	47	63	53	51	54	66	68	68	75	82	97	103	108	106	105	128	110	115		
1	131	109	143	170	175	148	166	152	143	154	151	161	170	181	122	120	121	123	122	122	122	119	184	202	137	138	127	106	106	106	106	106	106		
110	128	93	94	92	133	123	37	34	86	50	40	58	55	81	99	95	107	75	145	112	149	159	177	163	131	143	145	174	156	165	157	157			
07	156	92	80	68	60	42	43	57	51	58	72	60	66	80	60	51	47	64	59	89	116	85	124	125	135	100	12	65	73	43	55	64	51	79	94
187	182	182	185	180	118	120	120	119	119	118	121	120	118	147	209	174	148	90	64	79	75	73	66	81	86	80	55	78	77	79	72	72			
80	75	74	120	130	162	148	180	181	153	185	147	149	156	159	177	184	189	192	202	192	178	118	116	117	119	120	118	115	115	115	115	115			
77	72	73	71	77	87	114	123	110	109	85	135	92	35	20	86	97	41	47	62	72	74	90	98	152	132	134	134	149	107	149	165	141	168		
9	118	117	117	176	196	106	79	75	73	75	82	87	71	73	98	80	129	106	97	41	91	71	66	89	80	97	113	147	163	140	49	19	61	96	12
3	115	169	163	188	193	210	202	203	201	197	197	195	115	113	114	116	116	116	114	118	124	182	115	78	89	81	101	114	84	95	106	106	106		
147	69	92	95	94	96	116	61	43	75	65	100	101	155	97	113	124	102	121	90	150	139	174	175	198	202	202	194	188	187	160	173	173			
113	103	102	102	104	115	97	105	87	117	101	123	103	76	90	82	99	81	97	99	113	154	139	70	61	89	89	111	108	88	66	82	82	123	88	144
4	174	159	150	159	138	139	155	109	106	100	100	111	99	101	154	100	104	100	110	107	115	115	107	123	112	119	109	100	99	121	121	121			
08	104	82	64	114	114	121	108	143	174	157	155	142	120	111	111	107	123	150	179	177	137	138	129	128	142	134	145	131	141	51	51				
2	123	113	114	82	96	108	115	115	102	70	105	96	93	100	118	133	119	136	84	34	69	115	110	128	125	82	116	110	137	170	2				
143	119	137	144	130	135	124	18	25	25	25	58	131	107	106	119	129	128	135	139	132	120	125	87	93	100	98	128	139	132	132	132	132			
87	99	95	79	140	150	180	182	167	150	155	145	104	142	108	155	154	149	147	133	125	147	121	122	140	128	129	24	27	30	21	24	162			
15	101	94	108	95	141	167	124	130	121	108	135	116	115	141	140	142	116	94	94	102	121	122	132	110	131	151	166	163	173	156	156				
131	126	131	133	23	28	19	29	58	125	120	118	122	134	138	153	136	142	151	152	141	146	110	117	97	102	102	102	102	102	102	102	102			
2	175	177	174	164	151	130	134	151	172	180	126	90	134	73	90	147	160	155	131	144	118	127	121	131	123	129	29	36	42	56	52	156			
49	124	129	125	118	147	170	109	146	122	155	130	169	155	113	80	93	96	110	135	152	168	178	156	141	137	126	121	117	12	12	12				
128	126	121	119	122	89	94	91	115	122	116	139	132	133	138	140	146	138	135	150	153	165	149	115	137	149	158	179	172	183	183	183				
7	135	145	144	127	125	119	122	124	120	129	105	214	163	97	69	60	94	128	144	170	168	120	151	124	119	123	124	155	141	141	141				
0	161	178	170	132	154	126	150	170	175	128	71	53	48	50	93	162	145	134	106	114	109	120	114	136	125	115	141	116	133	131					
2	138	125	129	121	125	121	122	112	105	106	144	113	150	129	131	138	119	118	113	161	141	178	171	178	172	154	182	92	52	52					
14	118	119	120	122	125	128	127	124	127	136	197	219	157	54	73	81	79	129	141	139	193	166	118	137	120	121	119	120	99	99	99	99			
153	184	196	219	212	126	41	23	32	36	33	30	39	33	35	38	92	106	110	116	114	111	119	121	120	122	124	127	127	127	127	127				
2	130	127	136	119	118	97	101	92	138	111	126	10	114	124	97	121	121	14	171	193	160	174	155	222	236	166	38	37	32	32					
122	127	123	127	126	127	125	199	220	171	41	34	122	87	123	127	146	129	173	169	115	127	127	137	137	101	120	103	130	113						

Image as a vector of features

Images \leftrightarrow Vectors



60	60	255	255
60	60	255	255
60	60	255	255
128	128	128	128



60
60
255
255
60
60
255
255
60
60
255
255
128
128
128
128
128

Features and Targets

Each of our input images are 100x100 pixels

$$\mathbf{y} = \text{softmax}(\mathbf{W}^T \mathbf{x} + \mathbf{b})$$

Q: What will be the length of our input (feature) vectors \mathbf{x} ?

Q: What will be the length of our one-hot targets \mathbf{t} ?

Q: What are the shapes of \mathbf{W} and \mathbf{b} ?

Q: How many (scalar) parameters are in our model, in total?

Feature Mapping

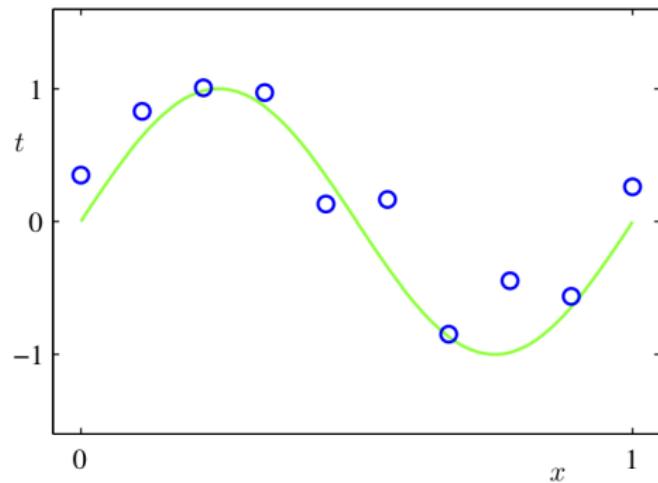
Computing New Features

In homework 2, we saw an example where computing **new features** could make a more powerful model.

- ▶ $x^{(1)} = -1, t^{(1)} = 1$
- ▶ $x^{(2)} = 1, t^{(2)} = 0$
- ▶ $x^{(3)} = 3, t^{(3)} = 1$

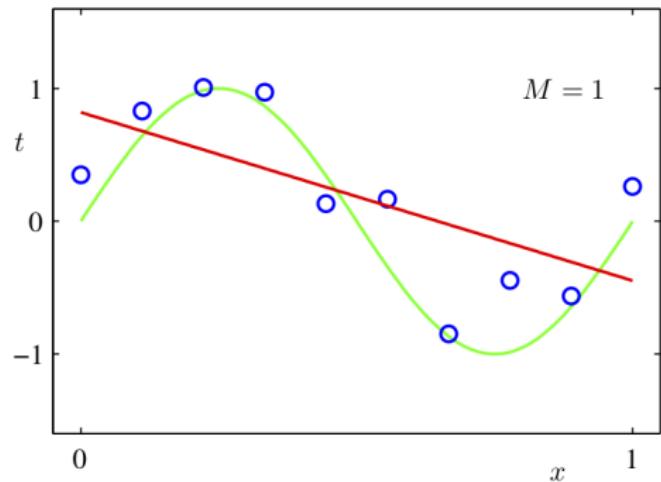
Example

Suppose we want to model the following data (from Bishop 2006):



Q: Will the model $y = wx + b$ fit the data well?

Linear Regression



Polynomial Feature mapping

One option to build a more powerful model is to fit a low-degree polynomial:

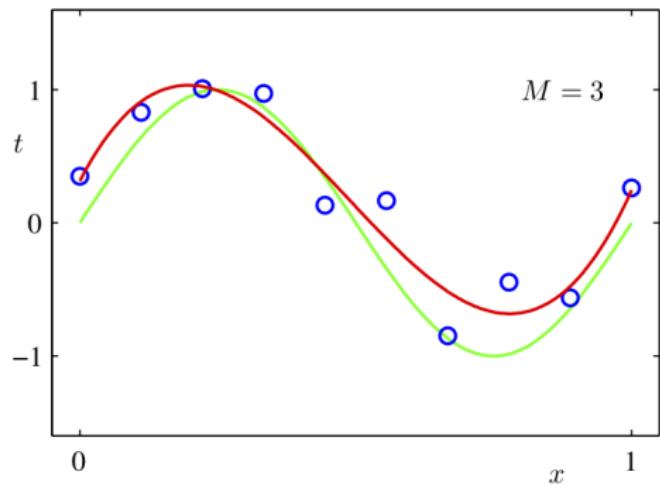
$$y = w_3x^3 + w_2x^2 + w_1x + w_0$$

The above model can still be framed as linear regression, by taking

$$\mathbf{x} = \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \end{bmatrix}$$

And we can find w_1 , w_2 , w_3 and $w_0 = b$ in the usual way.

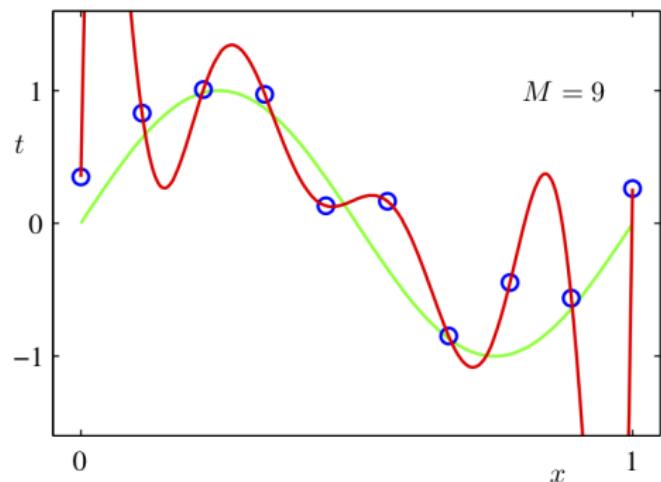
Fitting a degree 3 polynomial



Better fit!

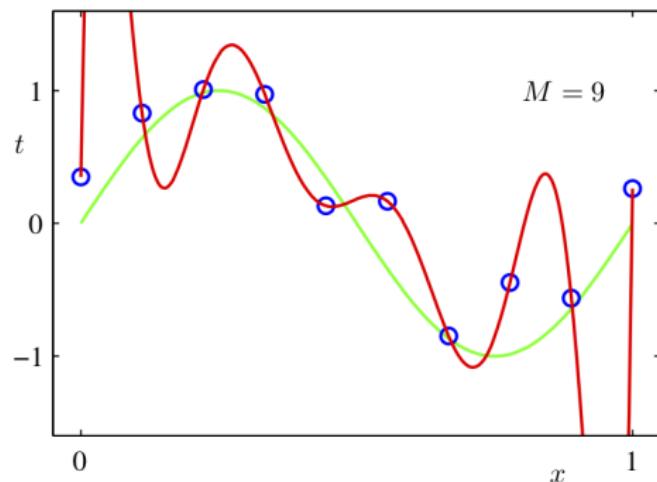
Higher degree polynomials

What about using an even higher degree polynomial?



Higher degree polynomials

What about using an even higher degree polynomial?



- ▶ This model fits the training data very well (cost = 0)
- ▶ ... but we don't expect this model to **generalize** to new data generated in the same way
- ▶ More parameter = more powerful model = more prone to **overfitting**

Feature Mapping in General

Computing the right features is very important. For example, if you want to predict whether someone will click on an ad for a machine learning book, you could compute:

- ▶ Last ad that they clicked on
- ▶ Last ad that they clicked on related to a book
- ▶ Time of day that this person is active
- ▶ How often this person clicks on ads

Q: How do you determine which features to include?

K-Nearest Neighbours

Same Example: Beatle Recognition

Given a 100x100 pixel colour image of a face of a Beatle, identify the Beatle



Four possible labels:

- ▶ John Lennon
- ▶ Paul McCartney
- ▶ George Harrison
- ▶ Ringo Starr

Linear Classification

We already saw today that we can frame the problem as a logistic regression problem

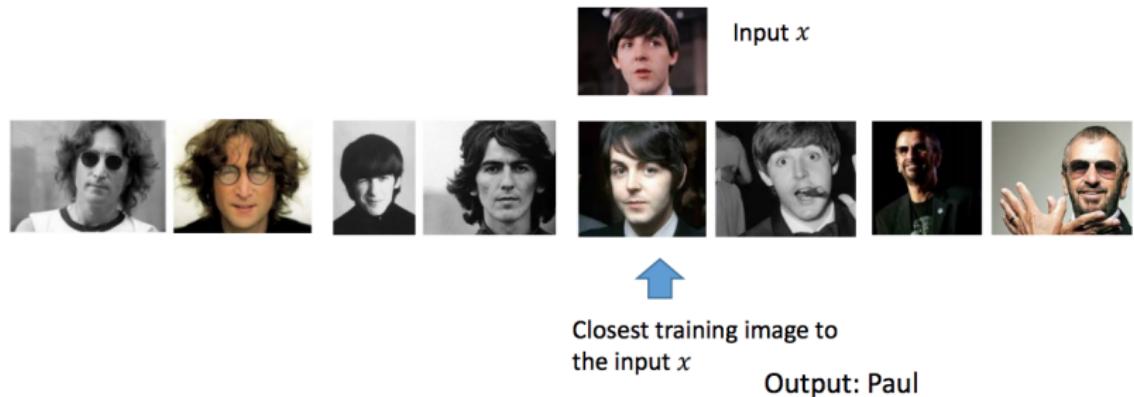
$$\mathbf{z} = \mathbf{Wx} + \mathbf{b}$$

$$\mathbf{t} = \text{softmax}(\mathbf{z})$$

Another approach: 1-nearest neighbour

For a new image x for which we want to make a prediction:

- ▶ Find the training photo/vector $x^{(i)}$ that is “closest” to x
- ▶ Output the prediction $y = t^{(i)}$



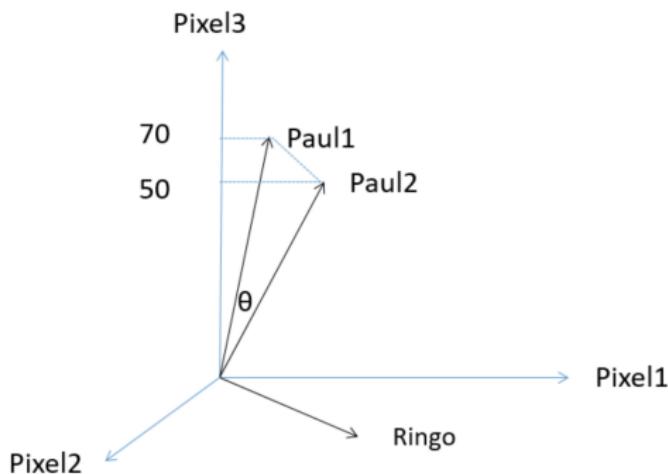
Are two images “close”?

Determining “closeness” using vector representations \mathbf{a} and \mathbf{b} of images

- ▶ Euclidean distance:

$$\|\mathbf{a} - \mathbf{b}\| = \sqrt{\sum_i (a_i - b_i)^2} = \sqrt{(\mathbf{a} - \mathbf{b})^T \cdot (\mathbf{a} - \mathbf{b})}$$

- ▶ Cosine distance: $\cos(\theta_{ab}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|}$



Distances Measure Choice

Which distance measure makes sense?

Depends on the *invariance* that you want:

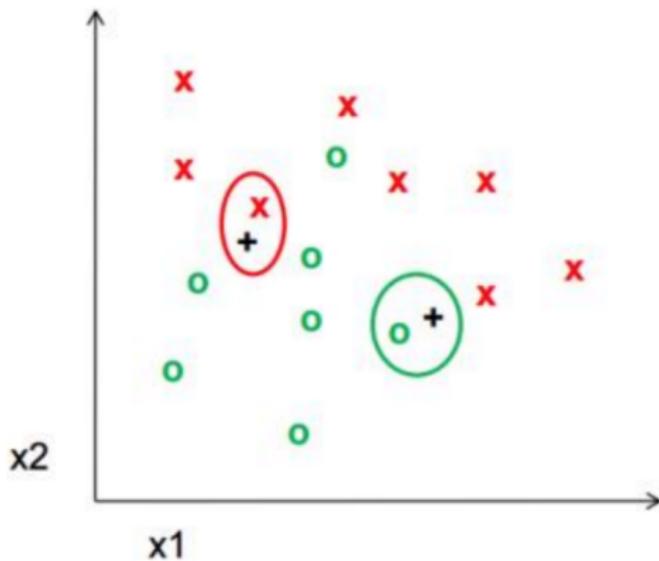
- ▶ Cosine distance is **scale invariant**: $dist(\mathbf{a}, \mathbf{b}) = dist(m\mathbf{a}, k\mathbf{b})$ for scalars m and K
- ▶ Euclidean distance is **shift invariant**:
 $dist(\mathbf{a}, \mathbf{b}) = dist(\mathbf{a} + \mathbf{c}, \mathbf{b} + \mathbf{c})$ for a vector \mathbf{c}

We'll use Euclidean distance in the next few slides, and cosine distance in project 1.

Example: 1-nearest neighbour

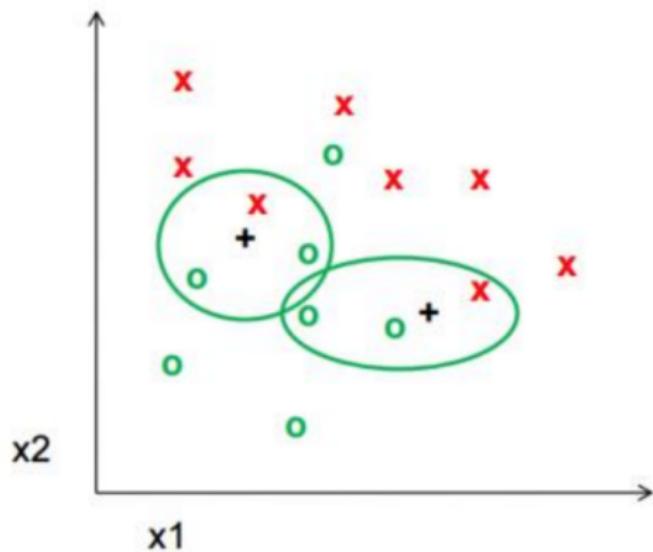
Task:

- ▶ Classify the new examples “+”
- ▶ Labels for the training set are GREEN and RED
- ▶ Choose Euclidean distance

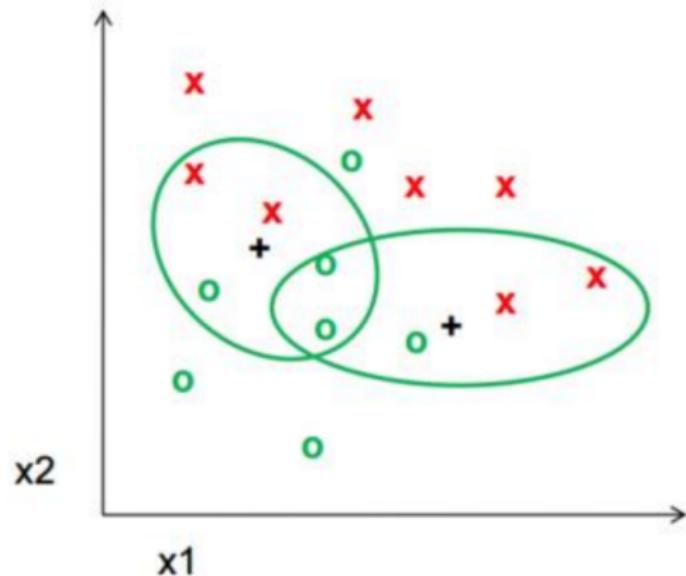


Example: 3-nearest neighbour

What if we use a larger set of neighbours?

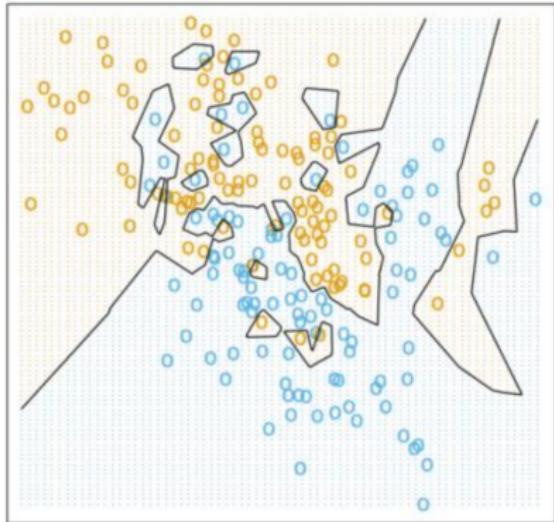


Example: 5-nearest neighbour

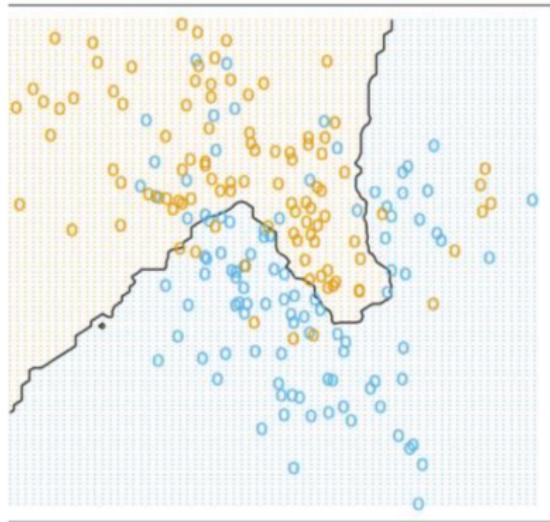


Choice of k

1-Nearest Neighbor Classifier



15-Nearest Neighbor Classifier



- ▶ If k is too small, then our model might be too “noisy”
 - ▶ Small change in x often changes the prediction
 - ▶ Model is prone to *overfitting*
- ▶ If k is too large, then our model might be too “simple”
 - ▶ Extreme example: $k = \text{size of training set}$

kNN vs Linear Models

These two families of models are *very* different!

- ▶ Linear models have linear **decision boundaries**, and kNN models have arbitrary decision boundaries
- ▶ Linear models have **parameters** (weights) that we choose via solving an optimization problem
- ▶ The k-Nearest Neighbour model requires the entire training data to be available to make predictions

Remaining questions

- ▶ How do we choose k ?
- ▶ How do we choose between different models?
- ▶ How do we know how well a model will perform on new data?

Generalization

Questions

- ▶ How do we choose k ?
- ▶ How do we choose which features to include?
- ▶ How do we choose between different models?
- ▶ **How do we know how well a model will perform on new data?**

The Training Set

The training set is used

- ▶ to determine the value of the **parameters**
- ▶ (in kNN) to make predictions

The model's prediction accuracy over the training set is called the **training accuracy**.

Q: Can we use the **training accuracy** to estimate how well a model will perform on new data?

The Training Set

The training set is used

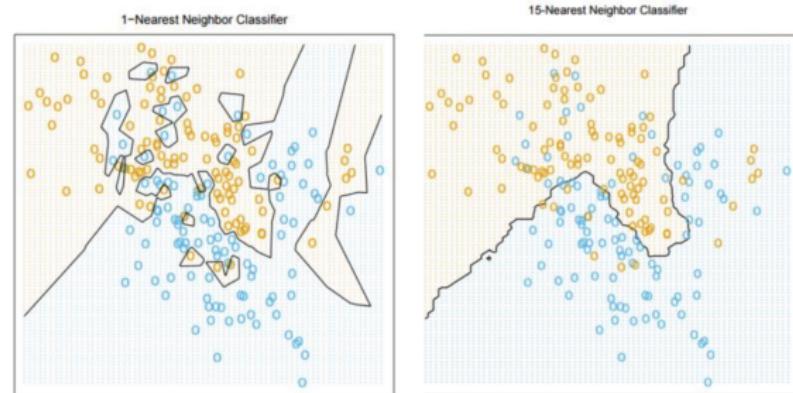
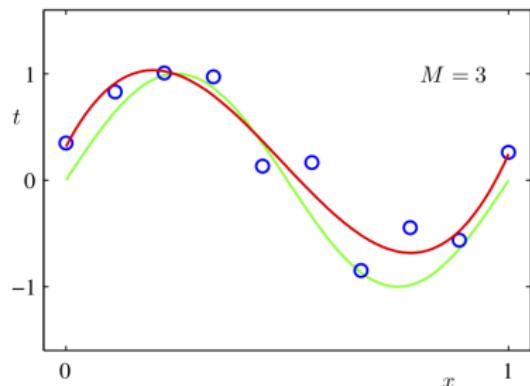
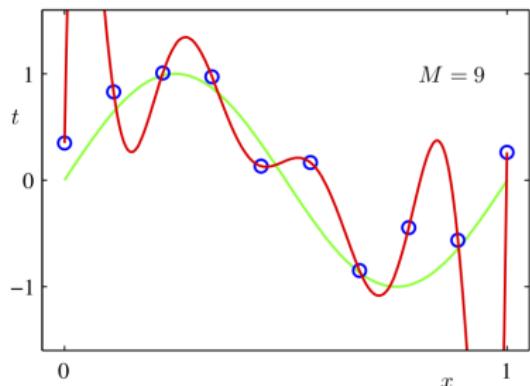
- ▶ to determine the value of the **parameters**
- ▶ (in kNN) to make predictions

The model's prediction accuracy over the training set is called the **training accuracy**.

Q: Can we use the **training accuracy** to estimate how well a model will perform on new data?

- ▶ No! It is possible for a model to fit well to the training set, but fail to *generalize*
- ▶ We want to know how well the model performs on *new data* that we didn't already use to optimize the model

Poor Generalization



Overfitting and Underfitting

Underfitting:

- ▶ The model is simple and doesn't fit the data
- ▶ The model does not capture *discriminative* features of the data

Overfitting:

- ▶ The model is too complex and does not generalize
- ▶ The model captures information about patterns in training set that happened by chance
 - ▶ e.g. Ringo happens to be always wearing a red shirt in the training set
 - ▶ Model learns: high red pixel content => predict Ringo

Preventing Overfitting

- ▶ Use a larger training set (expensive, often not feasible)
- ▶ Use a smaller network (requires starting over, might underfit)
- ▶ Other techniques (we'll explore later)

The Test Set

We set aside a **test set** of labelled examples.

The model's prediction accuracy over the test set is called the **test accuracy**.

The purpose of the test set is to give us a good estimate of how well a model will perform on new data.

Q: In general, will the test accuracy be *higher* or *lower* than the training accuracy?

Model Choices

But what about decisions like:

- ▶ Which k to use?
- ▶ Which model to use?

Q: Why can't we use the test set to determine which model we should deploy?

Model Choices

But what about decisions like:

- ▶ Which k to use?
- ▶ Which model to use?

Q: Why can't we use the test set to determine which model we should deploy?

- ▶ If we use the test set to make modeling decisions, then we will overestimate how well our model will perform on new data!
- ▶ We are “cheating” by “looking at the test”

The Validation set

We therefore need a third set of labeled data called the **validation set**

The model's prediction accuracy over the validation set is called the **validation accuracy**.

This dataset is used to:

- ▶ Make decisions about models that is **not continuous** and can't be optimized via gradient descent
- ▶ Example: choose k , choose which features x_j to use, choose α ,
...
 - ▶ These model settings are called **hyperparameters**
- ▶ The validation set is used to optimize **hyperparameters**

Splitting the data set

Example split:

- ▶ 60% Training
- ▶ 20% Validation
- ▶ 20% Test

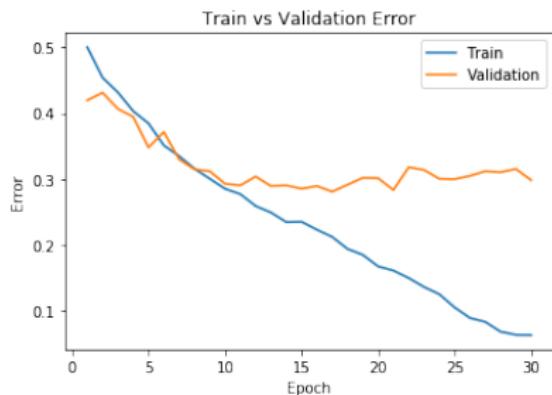
The actual split depends on the amount of data that you have.

If you have more data, you can get away with a smaller % validation and test.

Detecting Overfitting

Learning curve:

- ▶ **x-axis:** epochs or iterations
- ▶ **y-axis:** cost, error, or accuracy



Q: In which epochs is the model overfitting? Underfitting?

Q: Why don't we plot the test accuracy plot?

Strategies to Preventing Overfitting

- ▶ Collect more data: always the best first thing to try
- ▶ Use a simpler model: doesn't work well in practice
- ▶ Early-stopping: stop training before training accuracy convergences
 - ▶ In practice, save (or **checkpoint**) the weights after every E epochs. Use the weights that produce the highest validation accuracy
- ▶ Use a training strategy that reduces overfitting:
 - ▶ Example: weight decay

Weight Decay Idea

Idea: Penalize **large weights**, by adding a term (e.g. $\sum_k w_k^2$) to the cost function

Q: Why is it not ideal to have large (absolute value) weights?

Weight Decay Idea

Idea: Penalize **large weights**, by adding a term (e.g. $\sum_k w_k^2$) to the cost function

Q: Why is it not ideal to have large (absolute value) weights?

Because large weights mean that the prediction relies **a lot** on the content of one pixel (or one feature)

Weight Decay

- ▶ L^1 regularization: add a term $\sum_{j=1}^D |w_j|$ to the cost function
 - ▶ Mathematically, this term encourages weights to be exactly 0
- ▶ L^2 regularization: add a term $\sum_{j=1}^D w_j^2$ to the cost function
 - ▶ Mathematically, in each iteration the weight is pushed towards 0
- ▶ Combination of L^1 and L^2 regularization: add a term $\sum_{j=1}^D |w_j| + w_j^2$ to the cost function

Example: Weight Decay for Regression

Cost function:

$$\mathcal{E}(\mathbf{w}, b) = \frac{1}{2N} \sum_i ((\mathbf{w}\mathbf{x}^{(i)} + b) - t^{(i)})^2$$

Cost function with weight decay:

$$\mathcal{E}_{WD}(\mathbf{w}, b) = \frac{1}{2N} \sum_i ((\mathbf{w}\mathbf{x}^{(i)} + b) - t^{(i)})^2 + \lambda \sum_j w_j^2$$

Weight Decay Nomanclature

$$\mathcal{E}_{WD}(\mathbf{w}, b) = \frac{1}{2N} \sum_i ((\mathbf{w}\mathbf{x}^{(i)} + b) - t^{(i)})^2 + \lambda \sum_j w_j^2$$

$$\frac{\partial \mathcal{E}_{WD}}{\partial w_j} = \frac{\partial \mathcal{E}}{\partial w_j} + \lambda 2w_j$$

So the gradient descent update rule becomes:

$$w_j \leftarrow w_j - \alpha \left(\frac{\partial \mathcal{E}}{\partial w_j} + 2\lambda w_j \right)$$

Project 1 sklearn weight decay

- ▶ In project 1, you may get different answers from sklearn's logistic regression
 - ▶ Because of *stochastic* gradient descent
 - ▶ Because sklearn's logistic regression applies weight decay by default
- ▶ See https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegr

Reference

Some of these slides are based on the works of:

- ▶ Michael Guerzhoy
- ▶ Derek Hoiem
- ▶ Friedman, Hastie and Tibshirani
- ▶ Roger Grosse
- ▶ Bishop