Classification and regression

- What are the differences between these tasks?
- It seems to me that classification is much more useful and versatile than regression. Is this true?

Classification and regression

- Linear regression is mainly used for regression, but can also be used for classification.
- Logistic regression is mainly used for classification, but can also be used for regression.
  - Why is it still called “regression” then?

- What is the advantage of using logistic regression rather than linear regression for classification tasks?
  - Is linear regression used for classification in practice?
- Why is logistic regression not more popular for regression tasks?

Classification and regression

	Classification	Regression
Decision tree	Lec.1 (simplified form)	
k Nearest Neighbors	Lec.5	Possible
Perceptron	Lec.6	Possible
Linear regression	today	Lec. 6
Logistic regression	today!	
Neural networks	Next week	

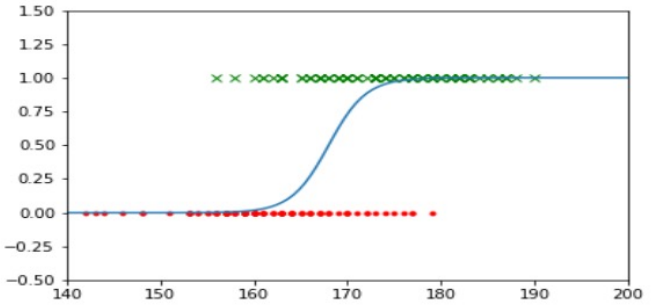
What can be used for regression?

- **Decision trees can't** properly handle regression by definition
  - Produce a limited number of judgments (good for classification, not so good for regression)
- **Logistic regression can** be used for regression
  - But not efficient and convenient
  - the underlying distributions are different (Bernoulli and Gaussian)

Linear models

- Logistic regression is said to be a linear classifier. Why is this so? The logistic function is non-linear, after all…
- What do we mean if we say ”best linear classifier”?
- Why does Marsland skip it if it is the best?

Logistic regression?



Interesting by itself

- A classifier
  - (not numerical regression)
- ”Standard” (”best”) purely linear classifier
- (Not in Marsland)

Useful tools for neural networks:

- The logistic function:
  - Its derivative
- Loss function
- Application of the chain rule for derivatives for gradient descent

Why is logistic regression linear?

- Logistic regression is a **linear** classifier, because this model is still **linear**
- Predicted values are expected to be determined by a **linear combination of the input variables (features)**
- It will find the optimal weights in the linear setup (given enough training steps)
- Also, comparatively easy to optimize (train)

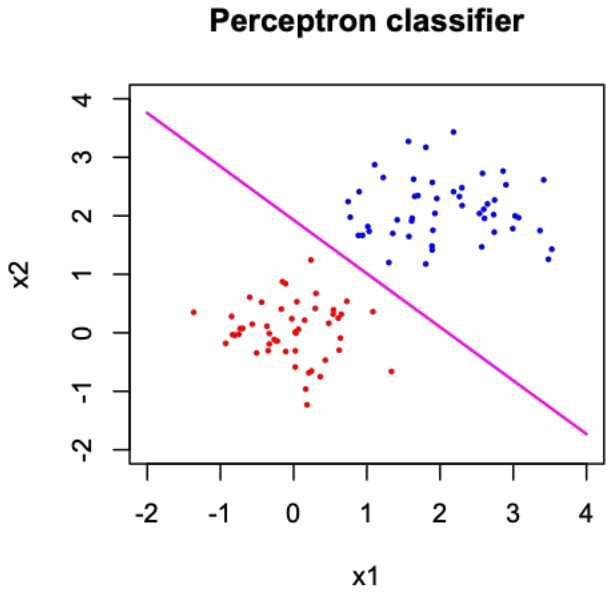
Do we need classifiers with linear boundaries?

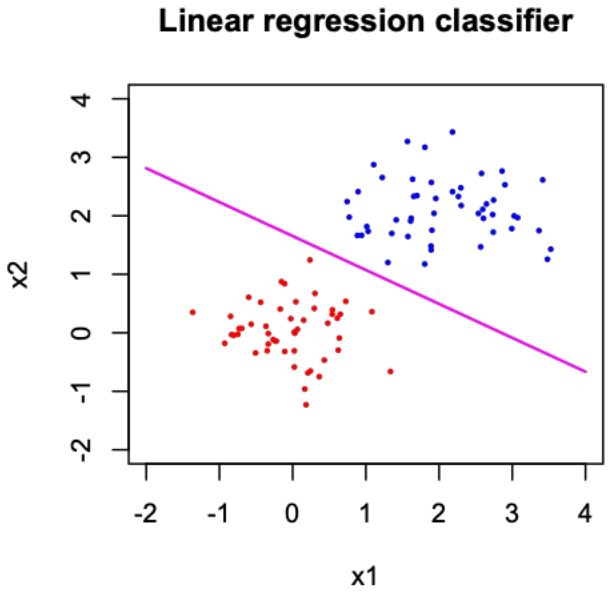
We have so far been introduced to (at least) four different classifiers:

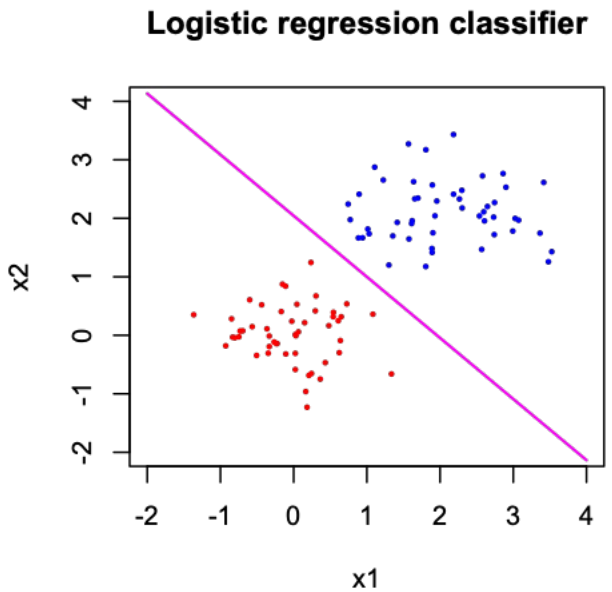
- k-nearest neighbor classifier] Nonlinear boundary between the two classes
- perceptron classifier
- linear regression classifier] Linear boundary between the two classes
- logistic regression classifier

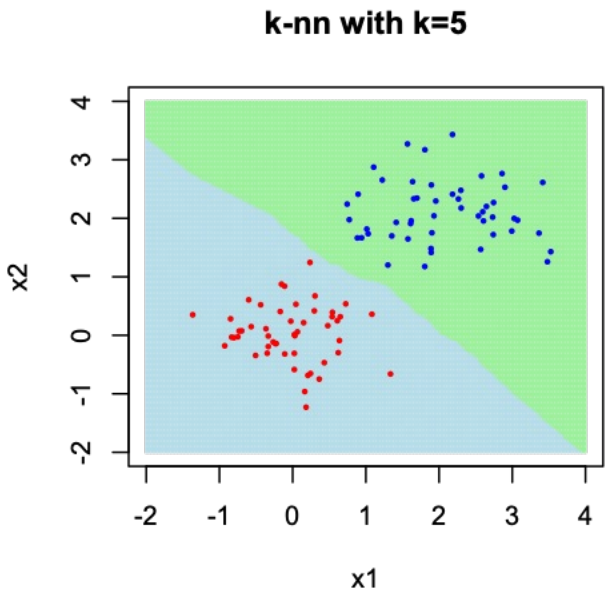
k-nearest neighbor is the only one that can handle both data that are linearly separable and data that are not linearly separable.

Linearly separable data

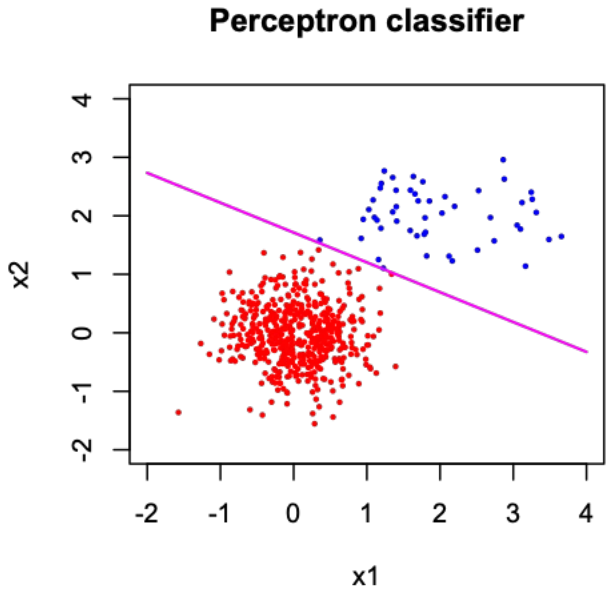


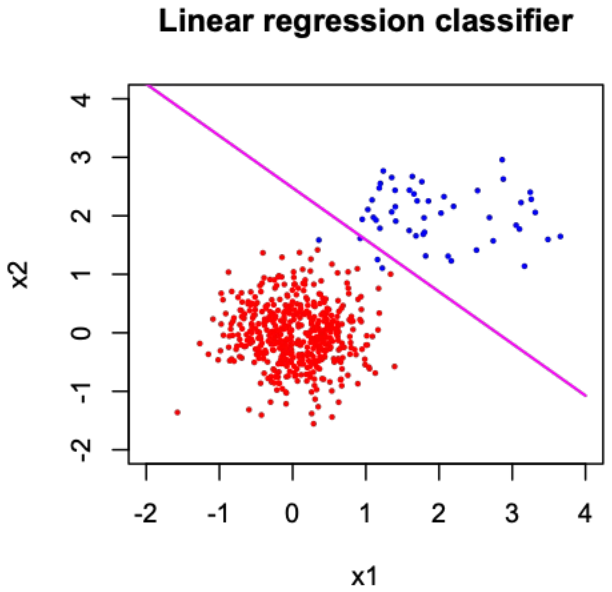


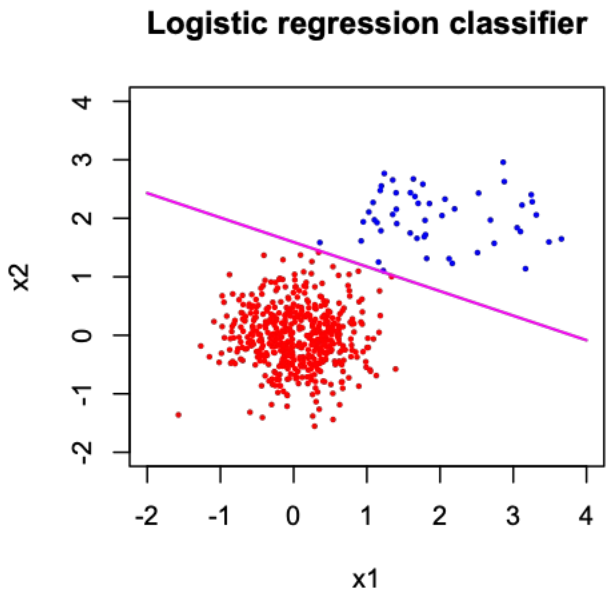


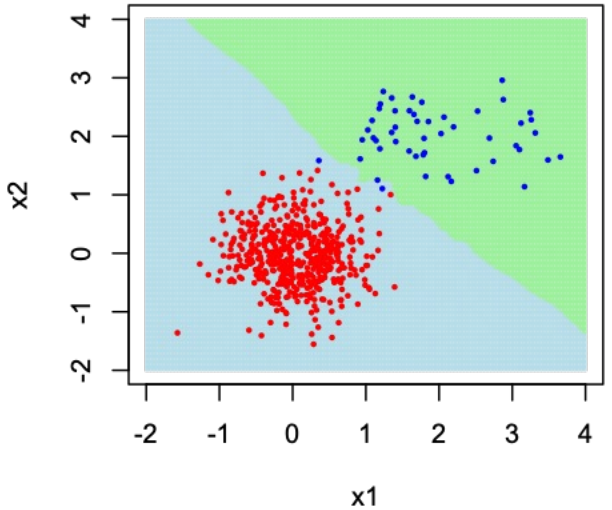


Almost linearly separable data

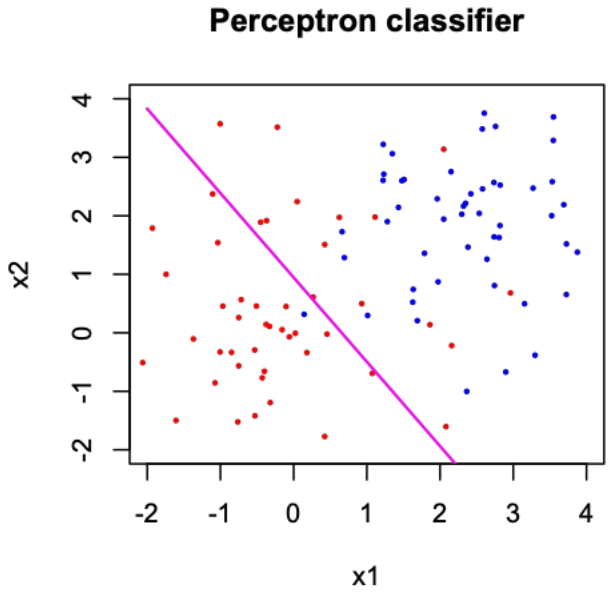


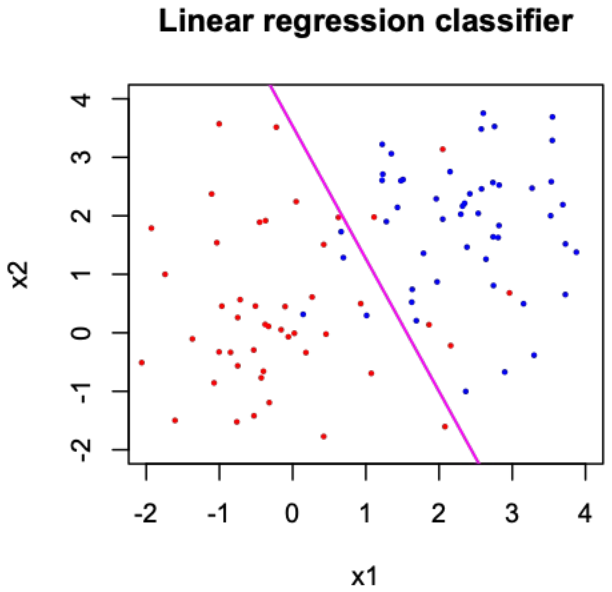


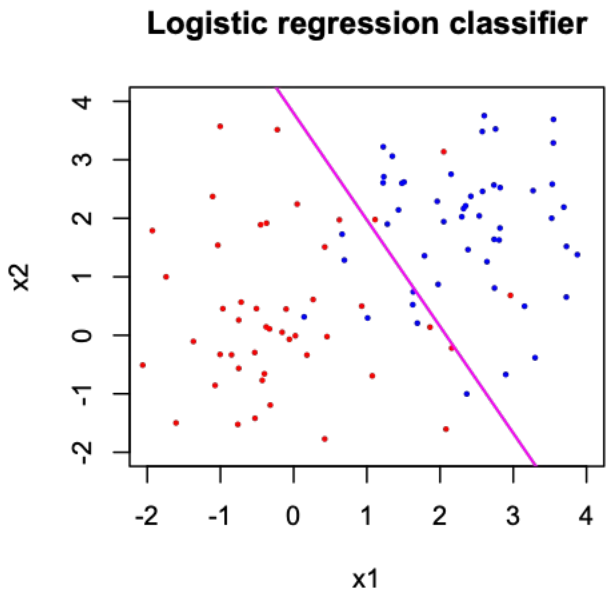


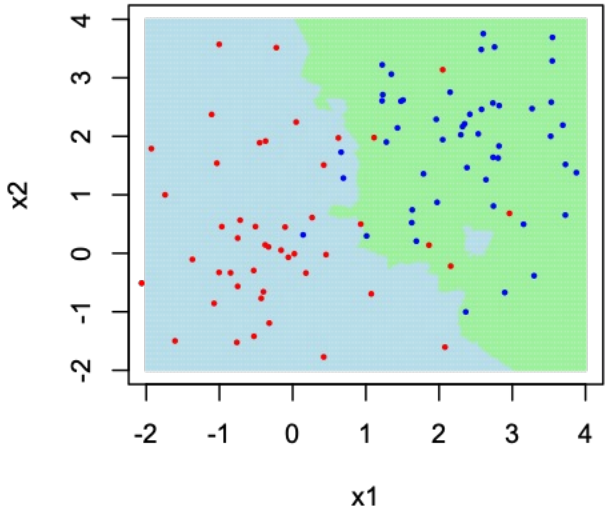


Partially linearly separable data

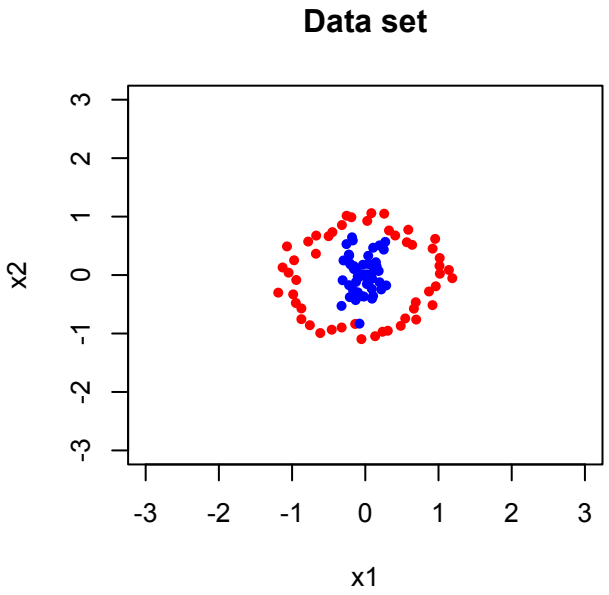


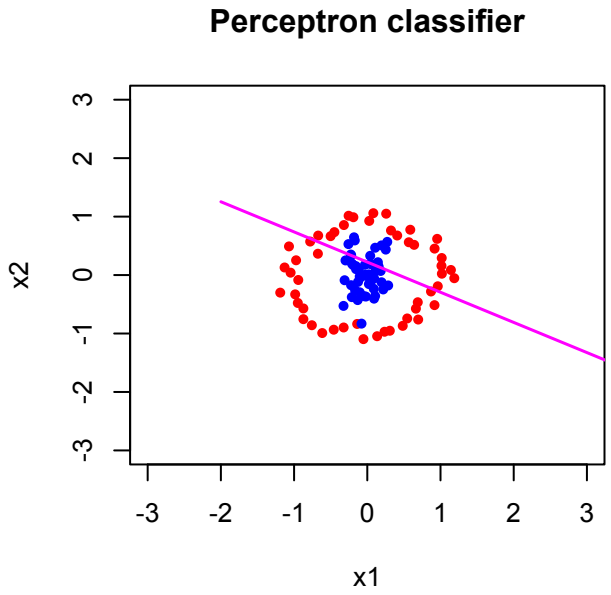


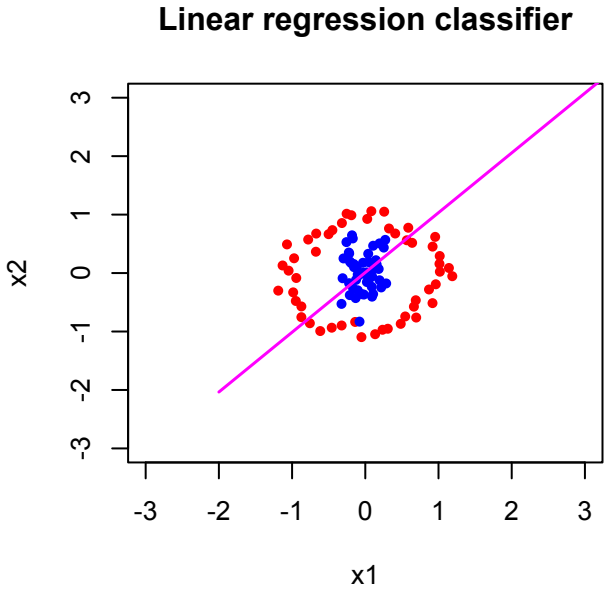


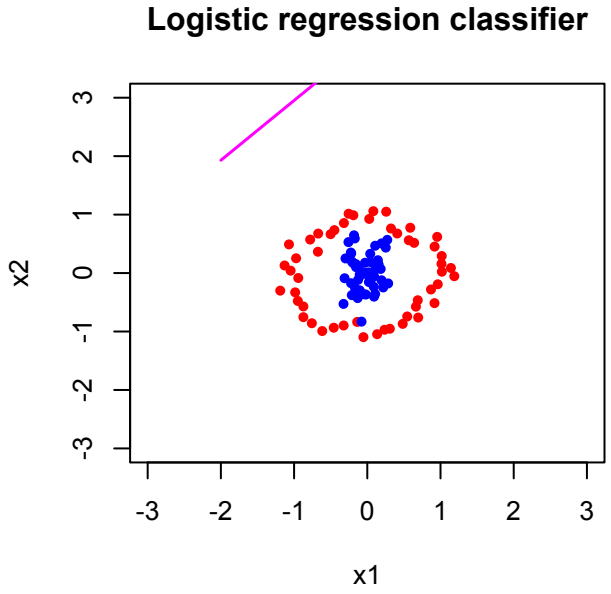


Not linearly separable data

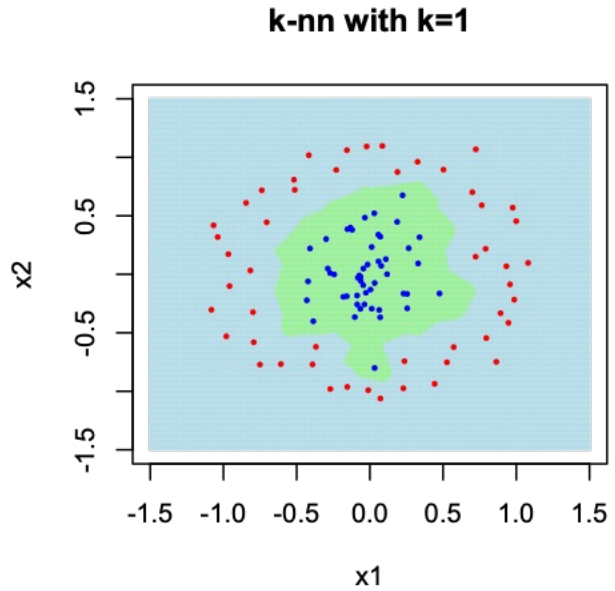


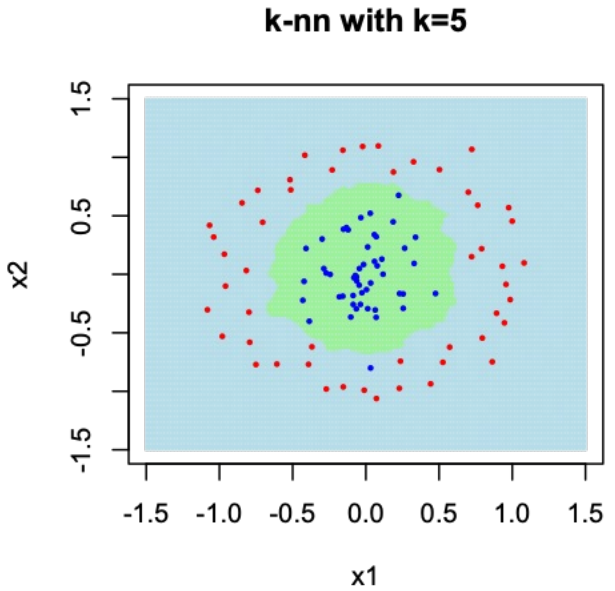


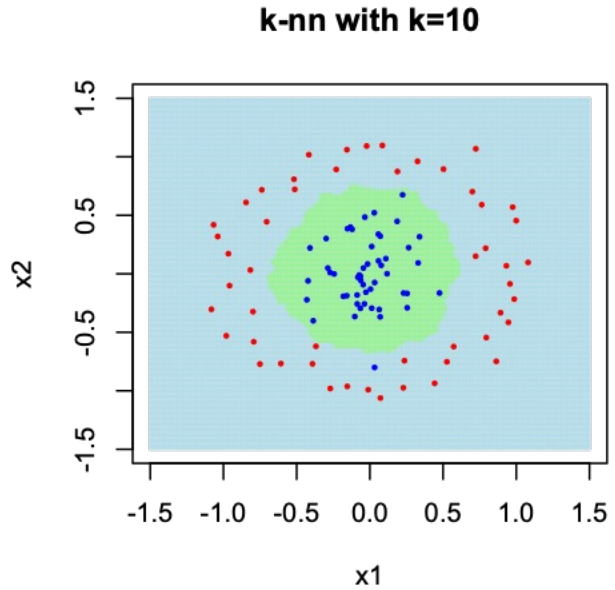


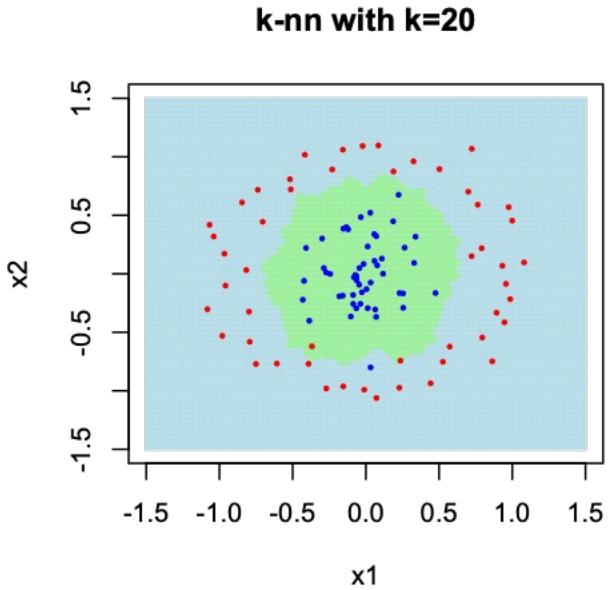


k-nn on linearly nonseparable data









Do we need classifiers with linear boundaries?

We have so far been introduced to (at least) four different classifiers:

- k-nearest neighbor classifier] Nonlinear boundary between the two classes
- perceptron classifier
- linear regression classifier] Linear boundary between the two classes
- logistic regression classifier

k-nearest neighbor is the only one that can handle both data that are linearly separable and data that are not linearly separable.

→ So why not use k-nn on all problems?

Logistic regression and perceptron

- These two classification models look very similar. What are the differences between them?

Model training

Imagine we have a labeled dataset and want to start training a model.

- How do we decide which type of model to use?
- Which variant of gradient descent should we use?
  - Batch training, mini-batch training, SGD
  - Can we use all these variants also with kNN, perceptron, linear regression?
- When does the training stop? Is 1 epoch enough?

Cross-entropy loss

The concept cross-entropy loss appears in the lectures.

Suppose we have:

- Input-output model: $y = f(\boldsymbol{x}; \boldsymbol{w})$
- Training data: $(\boldsymbol{x_i}, t_i)$, $\mathrm{i} = 1,2, \ldots, \mathrm{N}$

Two possible loss functions are:

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (t_i - f(\boldsymbol{x_i}; \boldsymbol{w}))^2$$

$$CE = -\sum_{i=1}^{N} [t_i \log f(\boldsymbol{x_i}; \boldsymbol{w}) + (1 - t_i) \log(1 - f(\boldsymbol{x_i}; \boldsymbol{w}))]$$

Cross-entropy loss

The concept cross-entropy loss appears in the lectures.

Suppose we have:

- Input-output model: $y = f(\boldsymbol{x}; \boldsymbol{w})$
- Training data: $(\boldsymbol{x_i}, t_i)$, $\mathrm{i} = 1,2, \ldots, \mathrm{N}$

Two possible loss functions are:

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (t_i - f(\boldsymbol{x_i}; \boldsymbol{w}))^2$$

$$CE = -\sum_{i=1}^{N} [t_i \log f(\boldsymbol{x_i}; \boldsymbol{w}) + (1 - t_i) \log(1 - f(\boldsymbol{x_i}; \boldsymbol{w}))]$$

→ **What is the difference here? What error does each of these actually measure?**
→ **For a given problem: How do we know which of these to choose?**

Difference between MSE and CE

- Mean squared error (MSE):
  - Measures the **scalar** difference between true values and model predictions
  - More suitable for regression-like problems
- Cross-entropy (CE):
  - Measures the difference between **probability distributions**
  - More suitable for classification-like problems

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Binary Cross-Entropy / Log Loss

MSE = mean squared error

n = number of data points

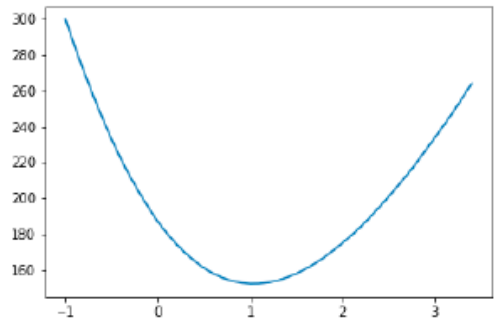
Y_i = observed values

$\hat{Y}_i$ = predicted values

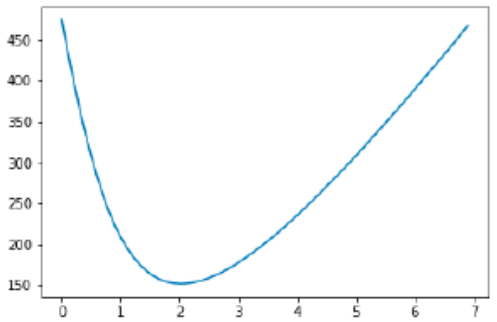
From this week's lectures:

Comparison

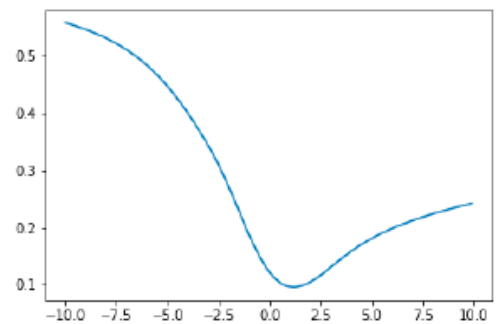
- The CE-loss is convex
  - The only minimum is global
- The MSE-loss is not convex when applied to logistic regression



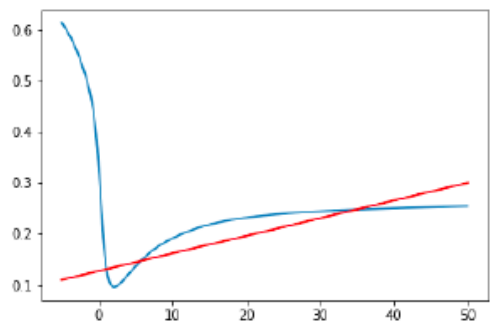
$\lambda w_1 Loss_{CE}(\boldsymbol{x}, t, (-2.51, w_1, 2.02))$



$\lambda w_2 Loss_{CE}(\boldsymbol{x}, t, (-2.51, 1.04, w_2))$



$\lambda w_1 Loss_{MSE}(\boldsymbol{x}, t, (-2.51, w_1, 2.02))$

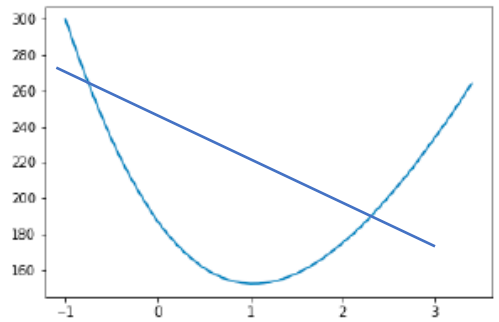


$\lambda w_2 Loss_{MSE}(\boldsymbol{x}, t, (-2.51, 1.04, w_2))$

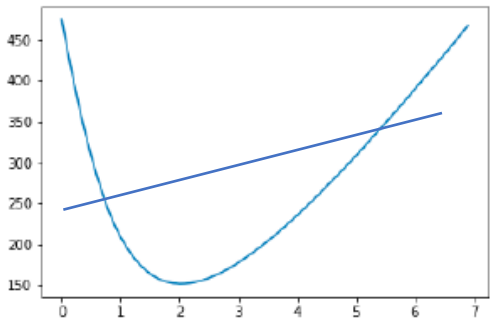
From this week's lectures:

Comparison

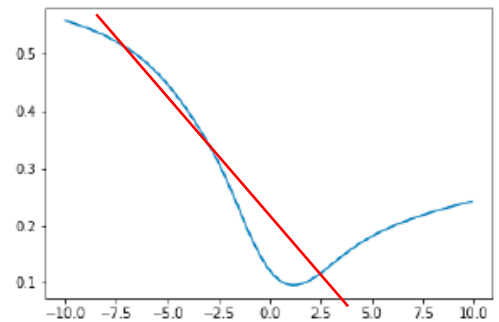
- The CE-loss is convex
  - The only minimum is global
- The MSE-loss is not convex when applied to logistic regression



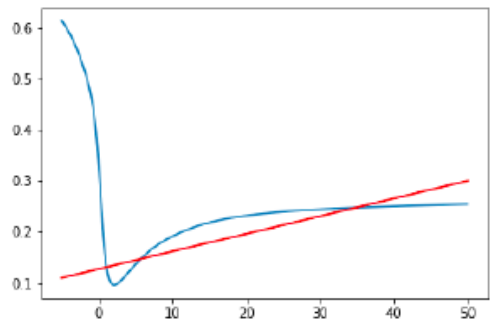
$\lambda w_1 Loss_{CE}(\boldsymbol{x}, t, (-2.51, w_1, 2.02))$



$\lambda w_2 Loss_{CE}(\boldsymbol{x}, t, (-2.51, 1.04, w_2))$



$\lambda w_1 Loss_{MSE}(\boldsymbol{x}, t, (-2.51, w_1, 2.02))$



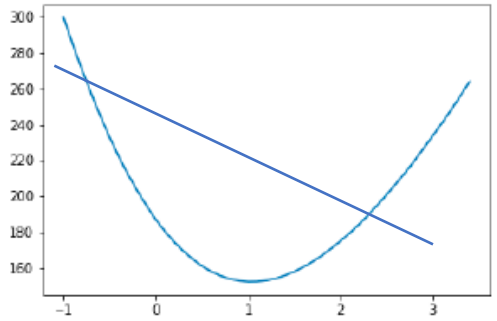
$\lambda w_2 Loss_{MSE}(\boldsymbol{x}, t, (-2.51, 1.04, w_2))$

From this week's lectures:

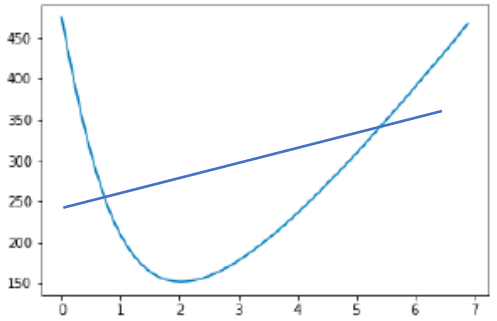
Comparison

- The CE-loss is convex
  - The only minimum is global
- The MSE-loss is not convex when applied to logistic regression

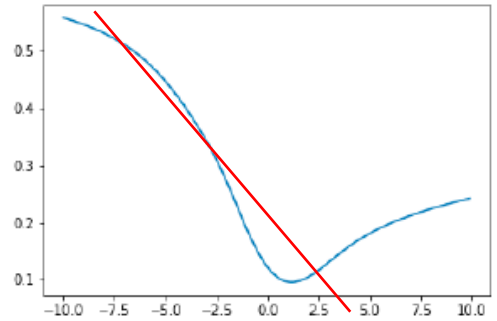
→ **What are the practical consequences of this lack of convexity?**



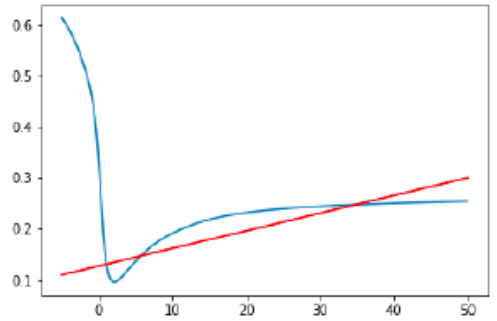
$\lambda w_1 Loss_{CE}(\boldsymbol{x}, t, (-2.51, w_1, 2.02))$



$\lambda w_2 Loss_{CE}(\boldsymbol{x}, t, (-2.51, 1.04, w_2))$



$\lambda w_1 Loss_{MSE}(\boldsymbol{x}, t, (-2.51, w_1, 2.02))$

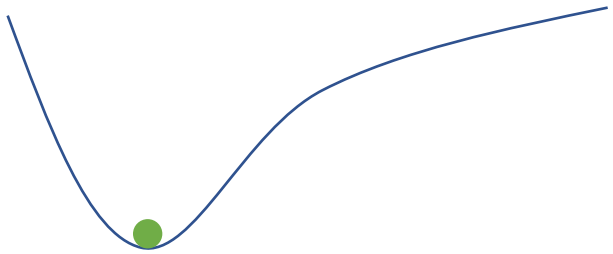


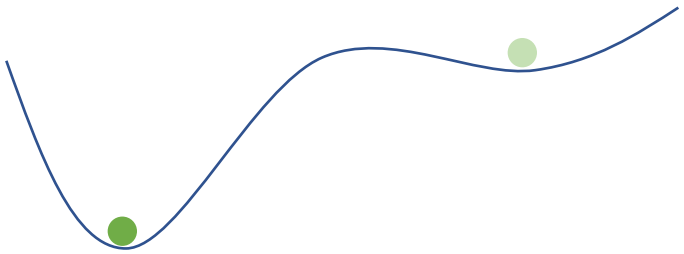
$\lambda w_2 Loss_{MSE}(\boldsymbol{x}, t, (-2.51, 1.04, w_2))$

Important theoretical result:

*If a loss function L(**w**) is strictly convex, then there is a single global minimum and no local minima.*

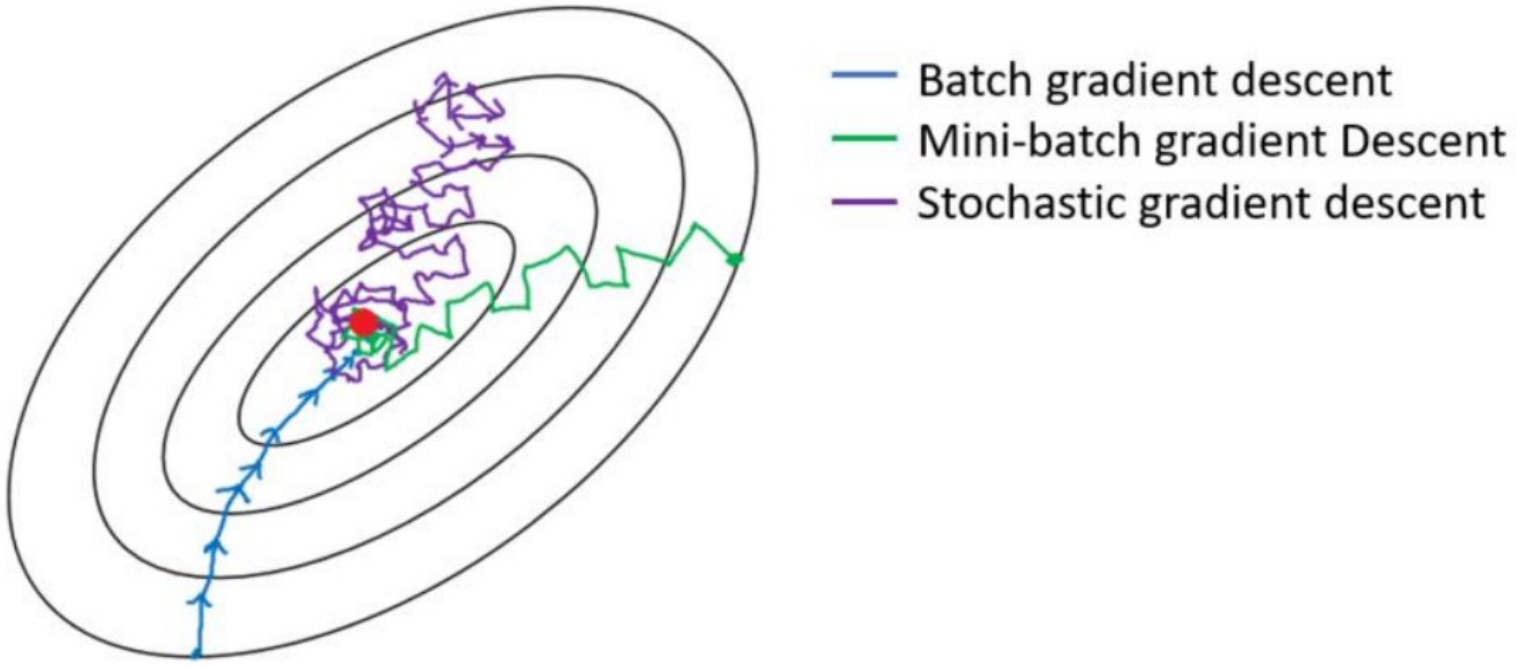
If it is not convex, then there *may* or *may not* be local minima.





Variants of gradient descent

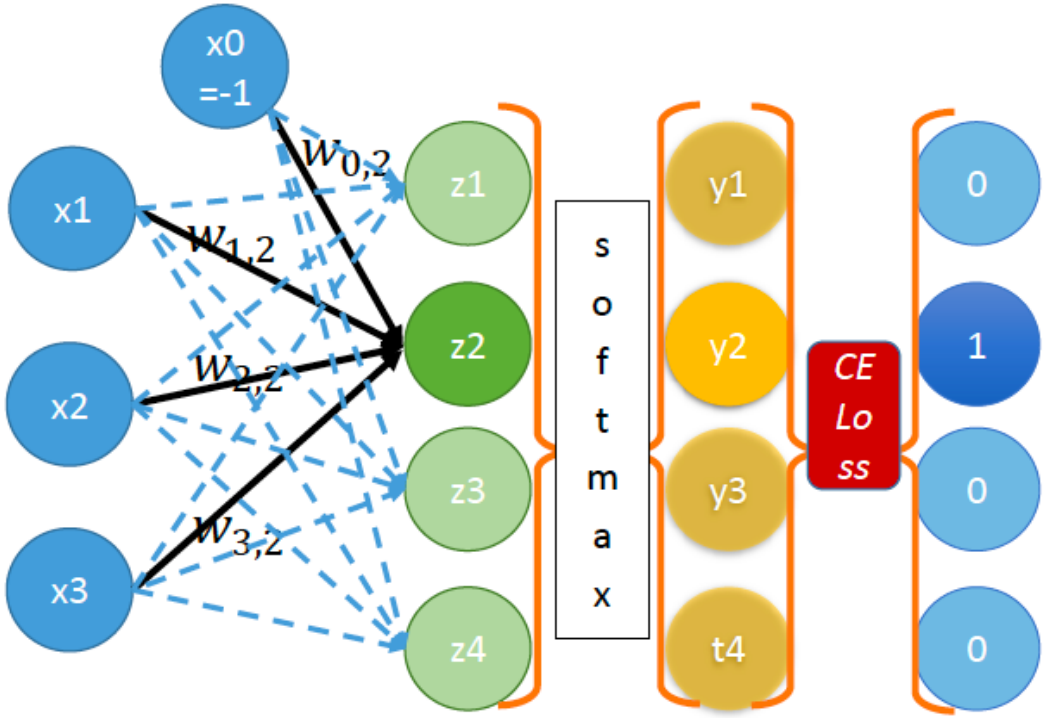
- Reminder: What is the difference between these? What are their pros/cons?
- Which one is applied most in practice? Why?



Multinomial regression:

- We can use this as a probabilistic classifier
  - $P(C_j|\vec{x}) = \frac{e^{\overrightarrow{w_j}\cdot\vec{x}}}{\sum_{k=1}^{n} e^{\overrightarrow{w_k}\cdot\vec{x}}}$
- To make hard decisions use

$$argmax_{j=1,\ldots,n} \frac{e^{\overrightarrow{w_j}\cdot\vec{x}}}{\sum_{k=1}^{n} e^{\overrightarrow{w_k}\cdot\vec{x}}}$$



→ Essentially logistic regression extended to more than two classes?
→ A neural network?