

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	

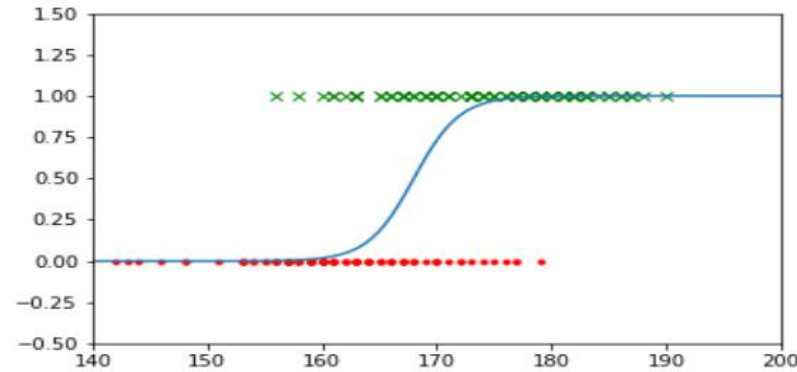
	Classification	Regression
Decision tree	Lec.1 (simplified form)	
k Nearest Neighbors	Lec.5	Possible
Perceptron	Lec.6	Possible
Linear regression	today	Lec. 6
Logistic regression	today!	
Neural networks	Next week	

- **Why do we call it linear/logistic regression** also when we use it for classification?
- Why don't we talk about logistic regression, neural networks and decision trees for regression?
- Are linear/logistic regression models used for classification **in practice**?

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)

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

The logistic function is not linear. Why do we call it a **linear** classifier?

What do we mean by it being the **best** linear classifier?

Why logistic regression is 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?

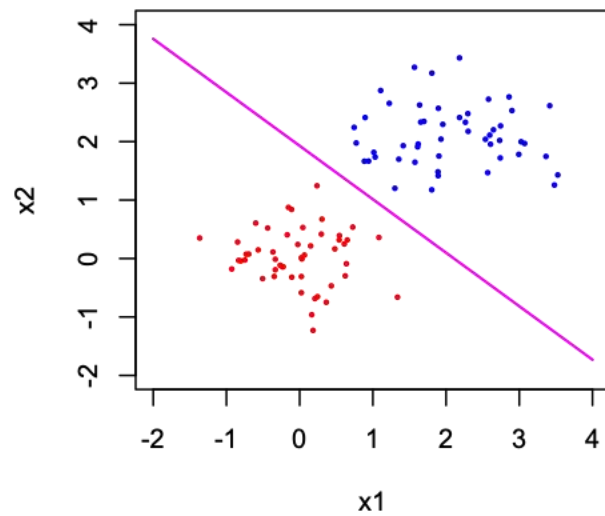
We have so far been introduced to (at least) four different classifiers:

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

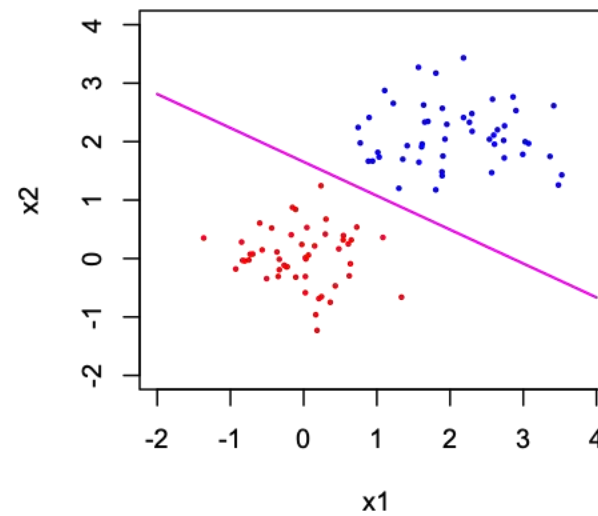
k-nearest neighbor is the only one that can handle both data that are linearly separable and data that are not linearly separable.

Linearly separable data

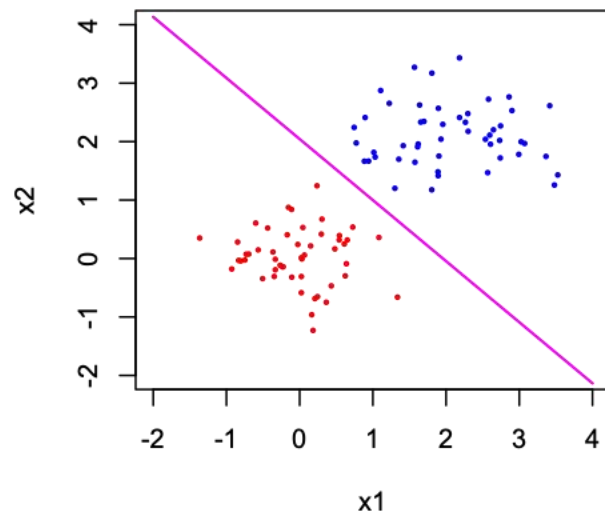
Perceptron classifier



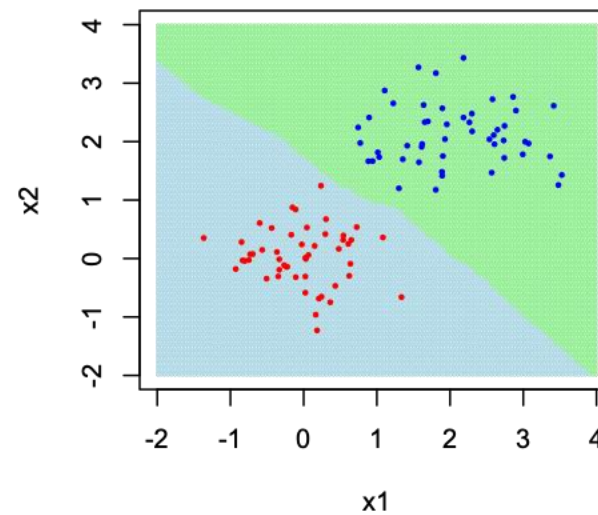
Linear regression classifier



Logistic regression classifier

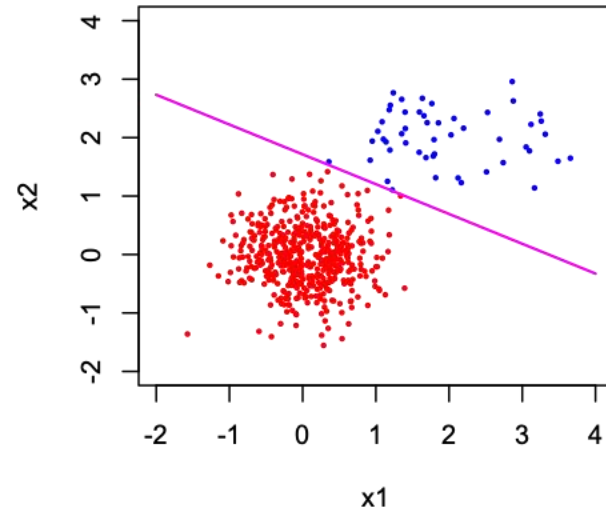


k-nn with k=5

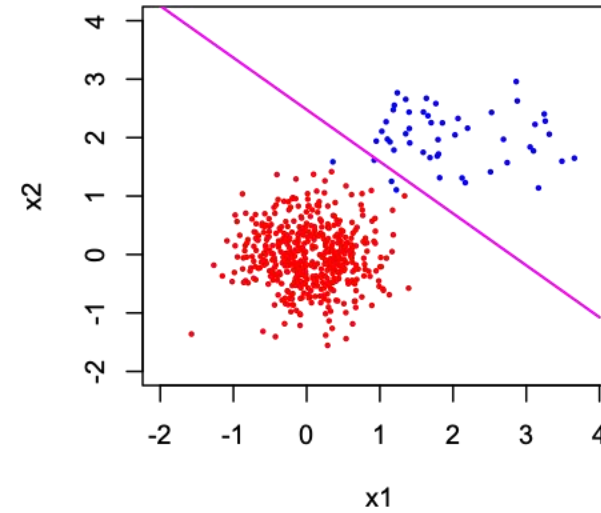


Almost linearly
separable data

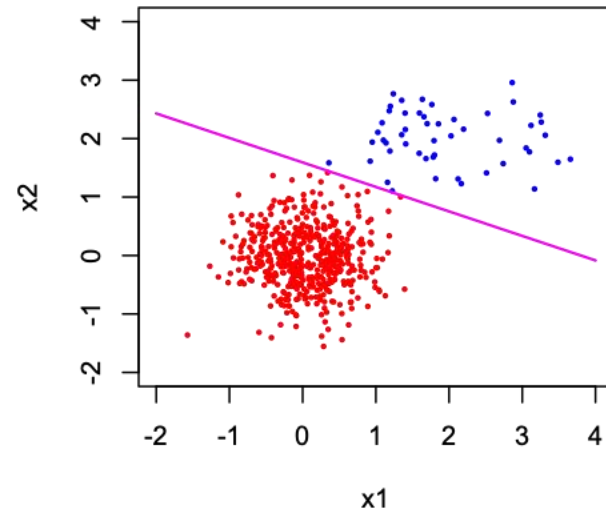
Perceptron classifier



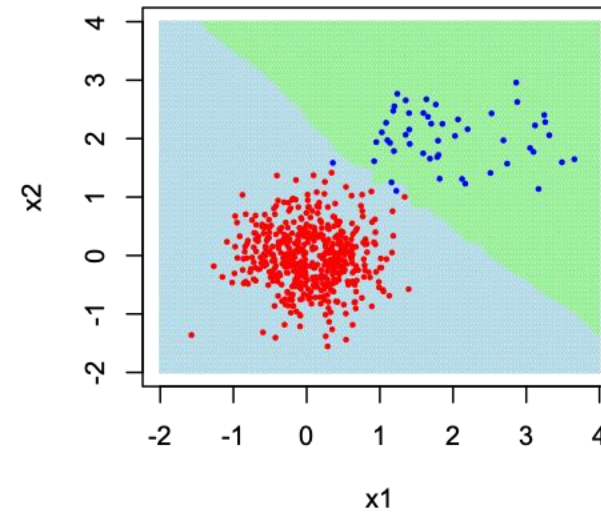
Linear regression classifier



Logistic regression classifier

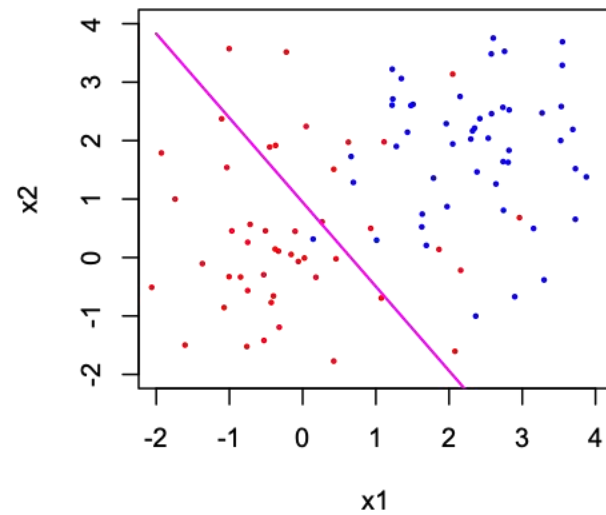


k-nn with k=5

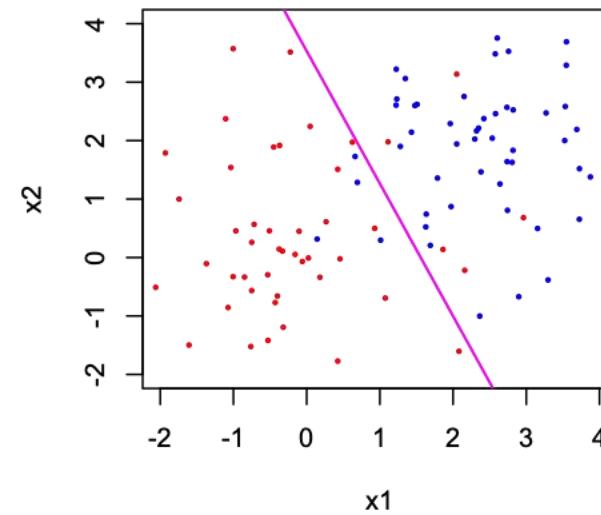


Partially linearly
separable data

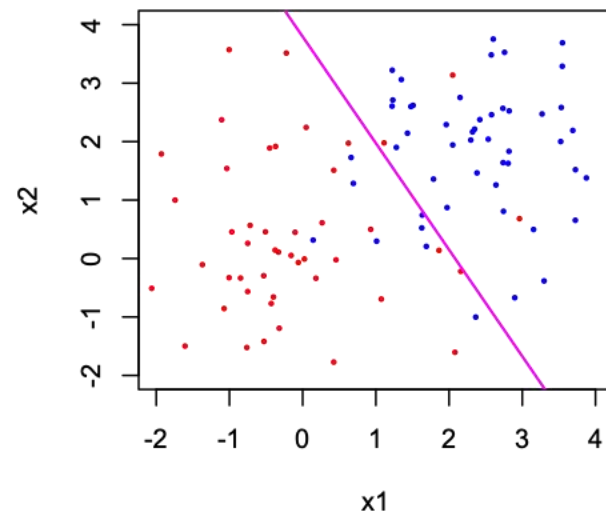
Perceptron classifier



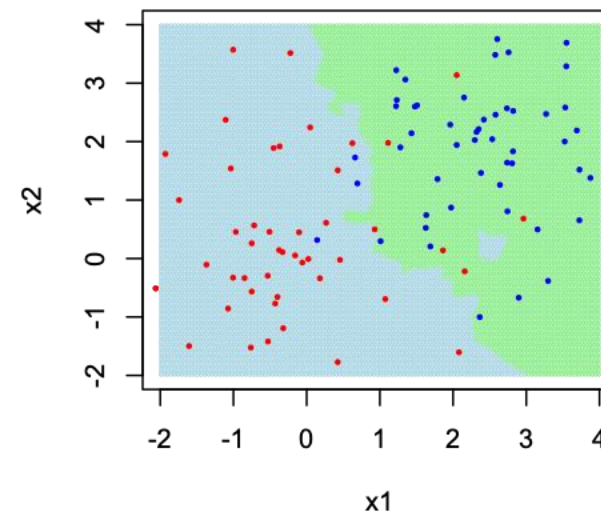
Linear regression classifier



Logistic regression classifier

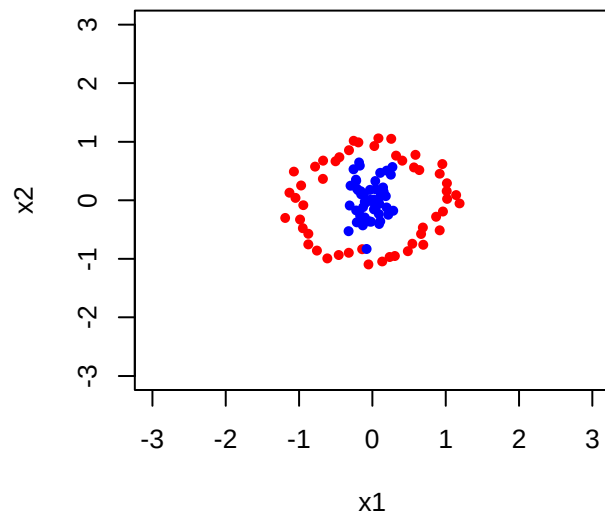


k-nn with k=5

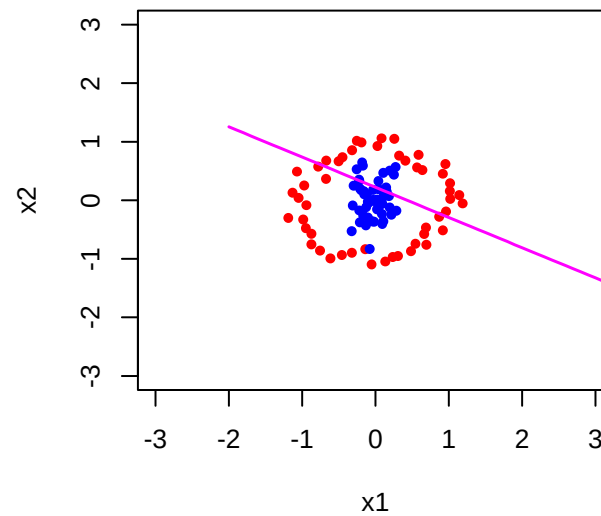


Not linearly
separable data

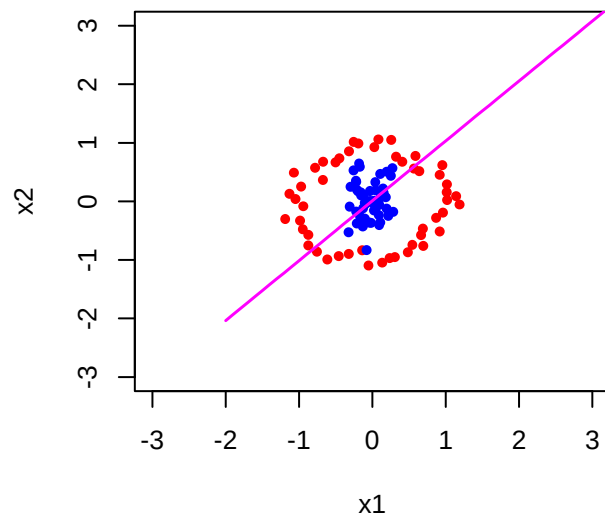
Data set



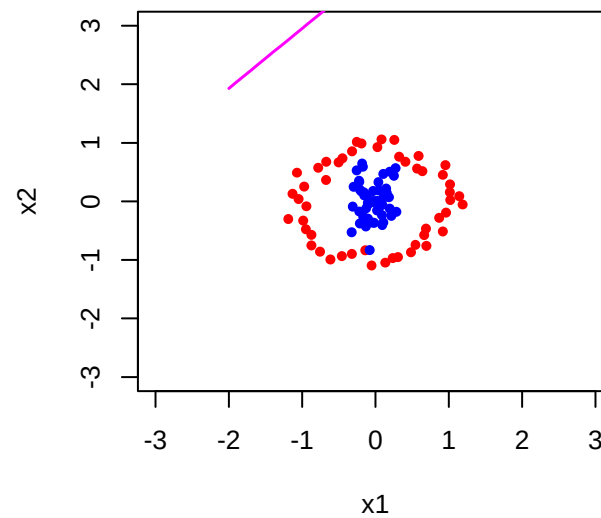
Perceptron classifier



Linear regression classifier

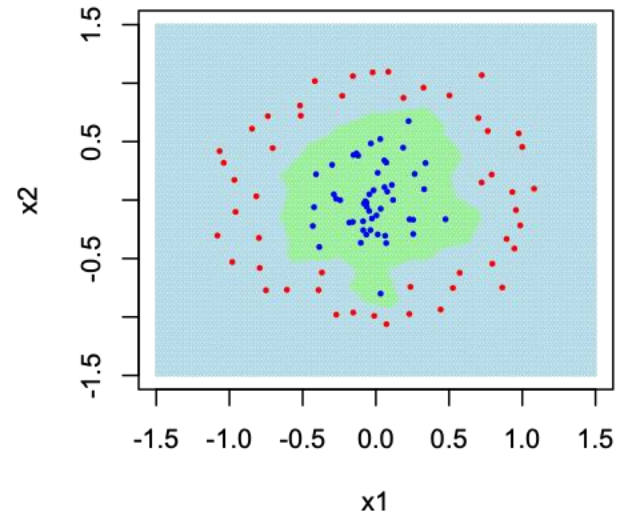


Logistic regression classifier

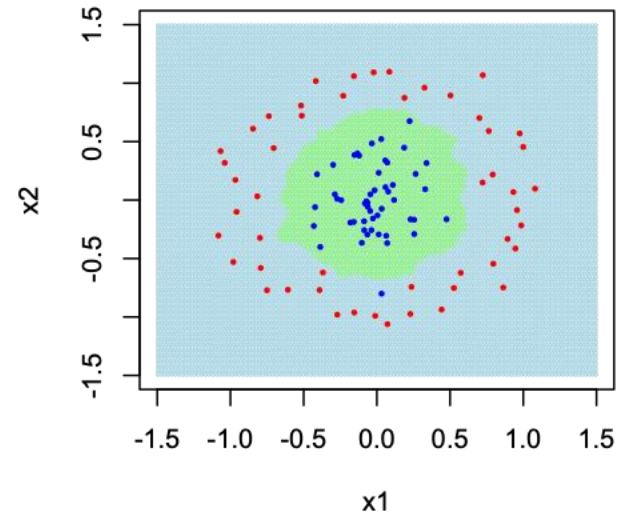


k-nn on linearly
nonseparable data

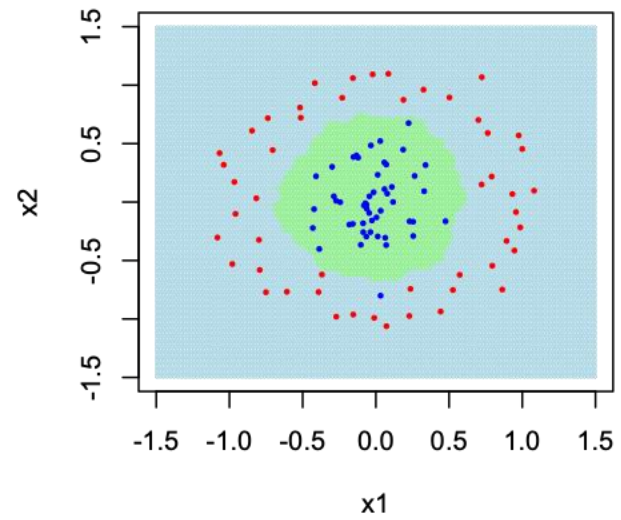
k-nn with k=1



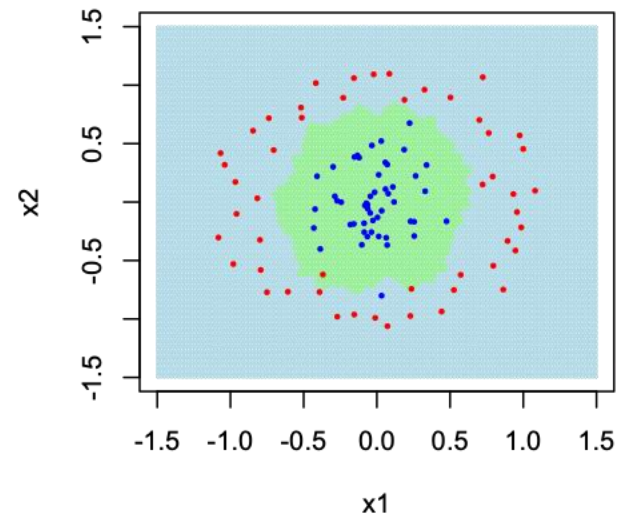
k-nn with k=5



k-nn with k=10



k-nn with k=20



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?

Cross-entropy loss

The concept **cross-entropy loss** appears in the lectures.

Suppose we have:

- Input-output model: $y = f(\mathbf{x}; \mathbf{w})$
- Training data: $(\mathbf{x}_i, t_i), \quad i = 1, 2, \dots, N$

Two possible loss functions are:

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

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

Cross-entropy loss

The concept **cross-entropy loss** appears in the lectures.

Suppose we have:

- Input-output model: $y = f(\mathbf{x}; \mathbf{w})$
- Training data: $(\mathbf{x}_i, t_i), \quad i = 1, 2, \dots, N$

Two possible loss functions are:

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

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

→ 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

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

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

MSE = mean squared error

n = number of data points

Y_i = observed values

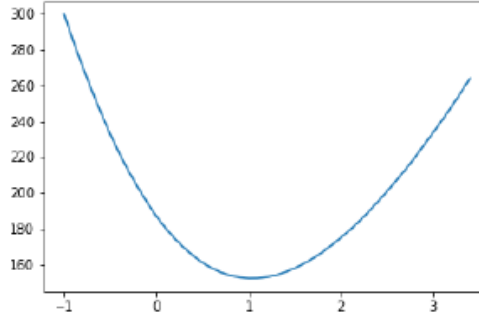
$\hat{Y}_i$ = predicted values

Binary Cross-Entropy / Log Loss

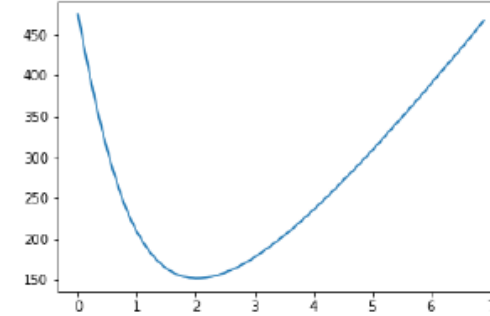
From this week's lectures:

Comparison

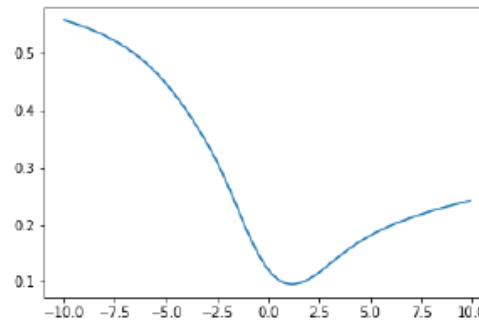
- The CE-loss is convex
 - The only minimum is global
- The MSE-loss is not convex when applied to logistic regression



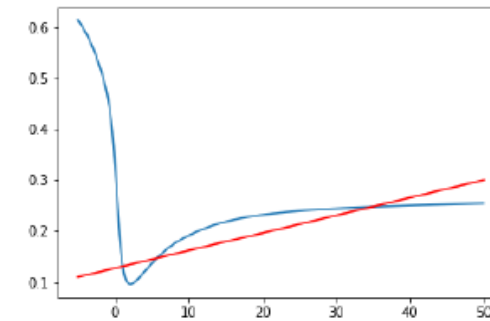
$$\lambda w_1 \text{Loss}_{CE}(\mathbf{x}, t, (-2.51, w_1, 2.02))$$



$$\lambda w_2 \text{Loss}_{CE}(\mathbf{x}, t, (-2.51, 1.04, w_2))$$



$$\lambda w_1 \text{Loss}_{MSE}(\mathbf{x}, t, (-2.51, w_1, 2.02))$$

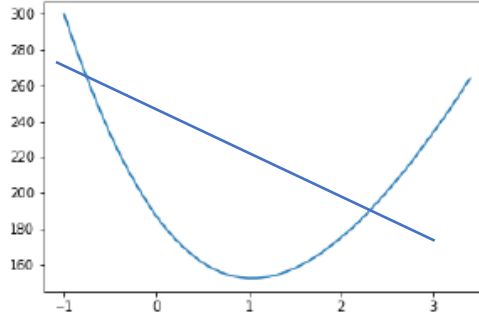


$$\lambda w_2 \text{Loss}_{MSE}(\mathbf{x}, t, (-2.51, 1.04, w_2))$$

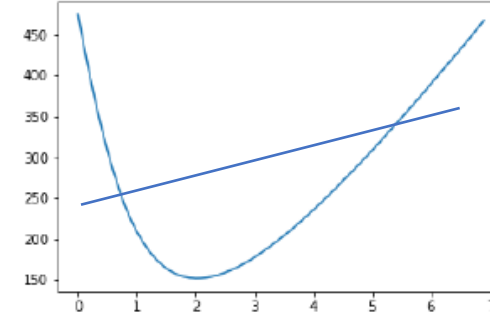
From this week's lectures:

Comparison

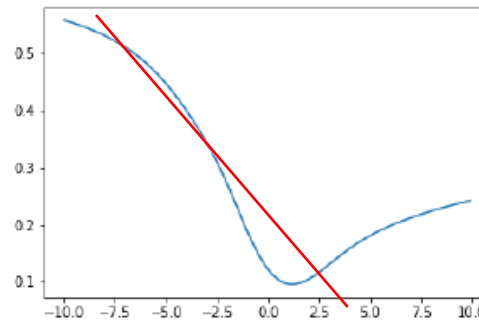
- The CE-loss is convex
 - The only minimum is global
- The MSE-loss is not convex when applied to logistic regression



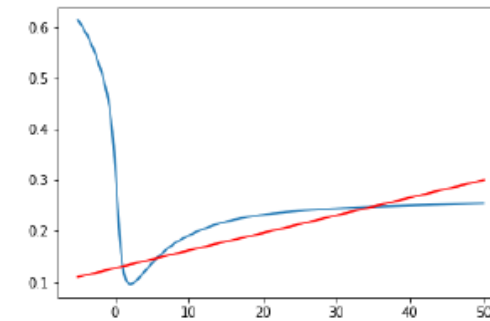
$$\lambda w_1 \text{Loss}_{CE}(\mathbf{x}, t, (-2.51, w_1, 2.02))$$



$$\lambda w_2 \text{Loss}_{CE}(\mathbf{x}, t, (-2.51, 1.04, w_2))$$



$$\lambda w_1 \text{Loss}_{MSE}(\mathbf{x}, t, (-2.51, w_1, 2.02))$$



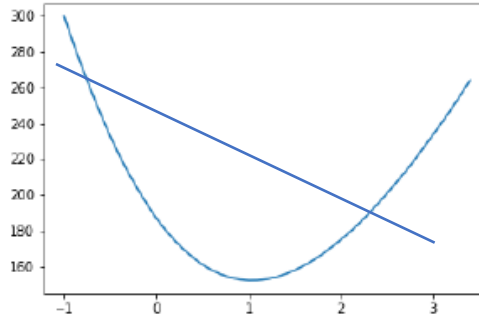
$$\lambda w_2 \text{Loss}_{MSE}(\mathbf{x}, t, (-2.51, 1.04, w_2))$$

From this week's lectures:

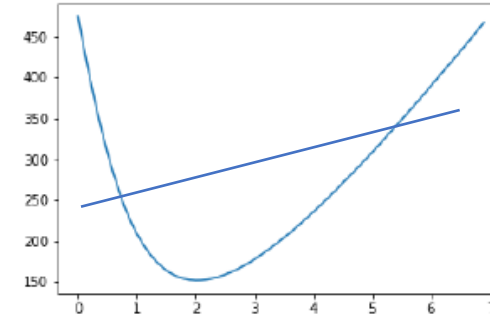
Comparison

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

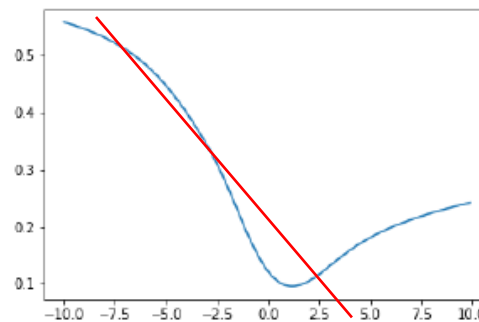
→ What are the practical consequences of this lack of convexity?



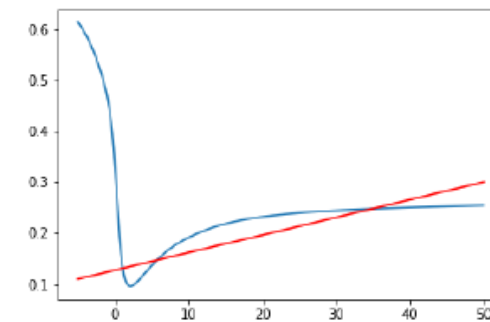
$$\lambda w_1 \text{Loss}_{CE}(\mathbf{x}, t, (-2.51, w_1, 2.02))$$



$$\lambda w_2 \text{Loss}_{CE}(\mathbf{x}, t, (-2.51, 1.04, w_2))$$



$$\lambda w_1 \text{Loss}_{MSE}(\mathbf{x}, t, (-2.51, w_1, 2.02))$$



$$\lambda w_2 \text{Loss}_{MSE}(\mathbf{x}, t, (-2.51, 1.04, w_2))$$

Important theoretical result:

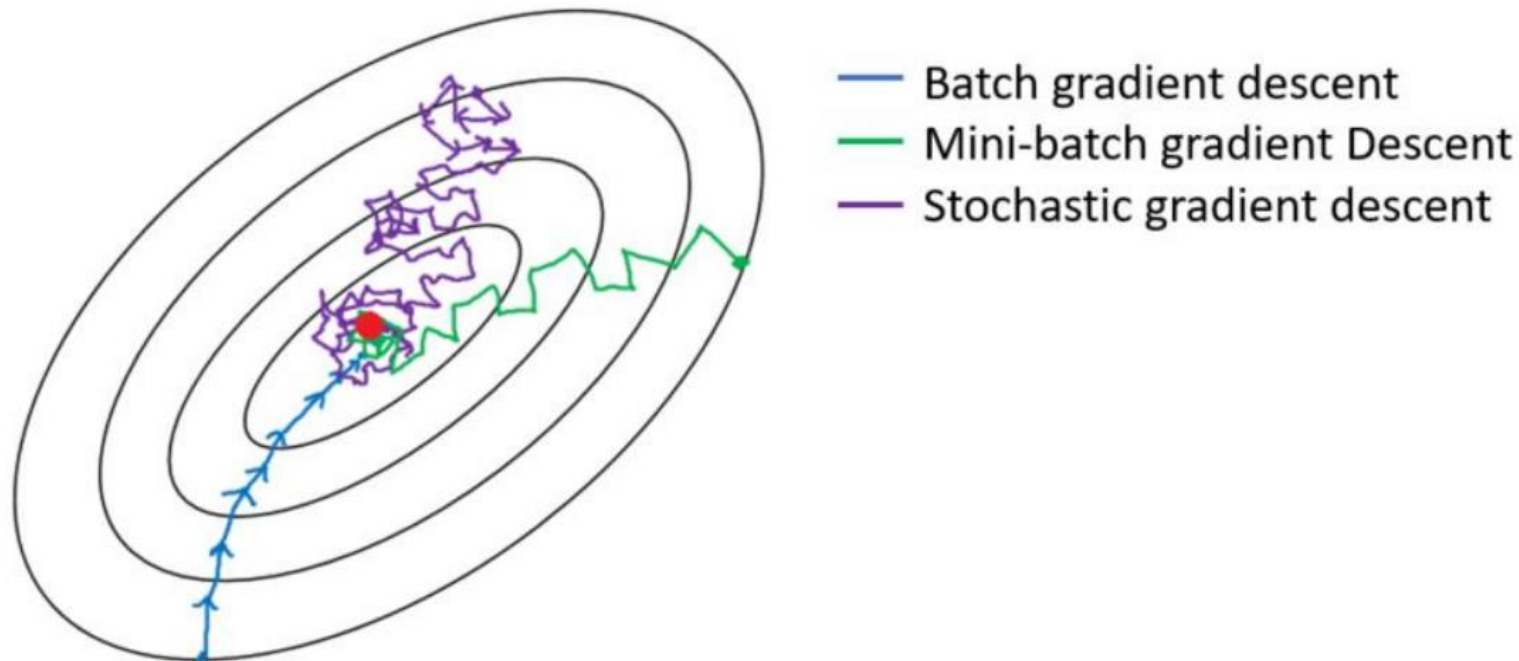
If a loss function $L(\mathbf{w})$ is strictly convex, then there is a single global minimum and no local minima.

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?



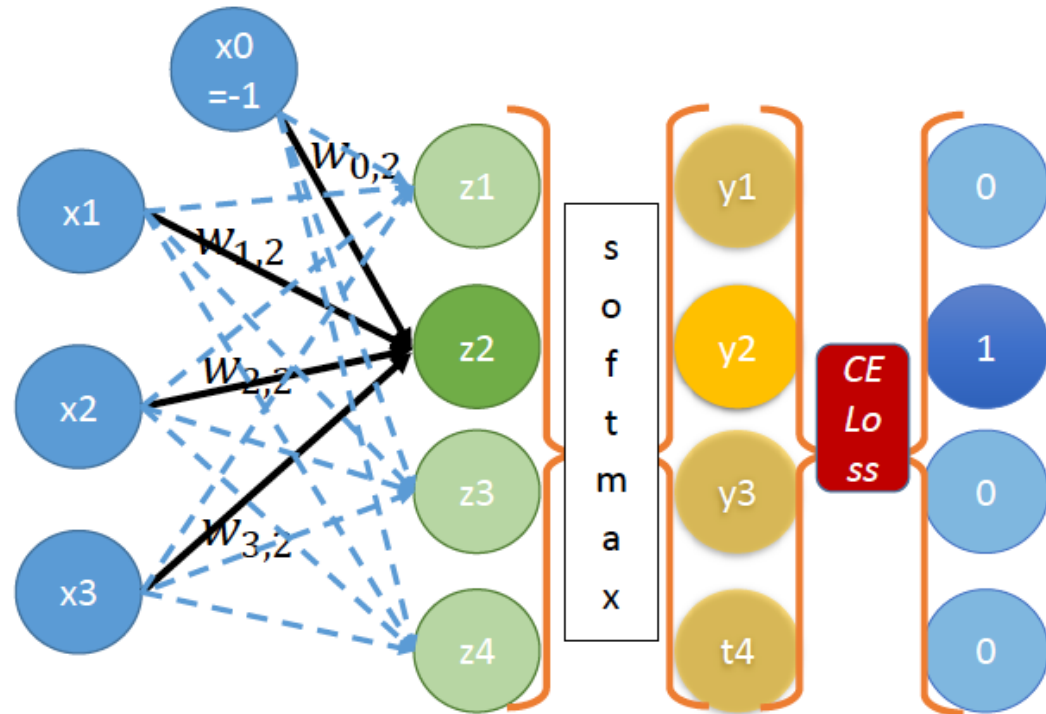
Multinomial regression:

- We can use this as a probabilistic classifier

- $P(C_j|\vec{x}) = \frac{e^{\vec{w}_j \cdot \vec{x}}}{\sum_{k=1}^n e^{\vec{w}_k \cdot \vec{x}}}$

- To make hard decisions use

- $\operatorname{argmax}_{j=1,\dots,n} \frac{e^{\vec{w}_j \cdot \vec{x}}}{\sum_{k=1}^n e^{\vec{w}_k \cdot \vec{x}}}$



→ Essentially logistic regression extended to more than two classes?

→ A neural network?