

quiz

convex optimization

sp1

A function f is continuous differentiable, that is $f \in C^1$, if

Vælg en eller flere:

- ☒ f has continuous first order partial derivatives
- ☐ f is differentiable
- ☒ f is differentiable and its derivative is continuous ✓
- ☐ f has first order partial derivatives
- ☐ f is continuous and its partial derivatives exists

sp2

The gradient ∇f of a differentiable function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is

Vælg en eller flere:

☒ a scalar if $n = 1$ ✓

☐ a vector with coordinates equal to the second order partial derivatives of f

☐ a scalar

☐ a square $n \times n$ matrix

☒ a vector with coordinates equal to the first order partial derivatives of f ✓

☒ the derivative of f

☒ a n -vector ✓

sp3

Let $x \in \mathbb{R}^n$. The Hessian $H(x)$ of a C^2 function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is

Vælg en eller flere:

☒ a square $n \times n$ matrix ✓

☒ a scalar if $n = 1$

☐ a vector with coordinates equal to the second order partial derivatives of f

☐ a n -vector

☐ a scalar

☒ the Jacobian matrix of ∇f ✓

☐ a vector with coordinates equal to the first order partial derivatives of f

☐ the Jacobian matrix of f

☒ a symmetric matrix ✓

☒ the second derivative of f ✓

☐ the derivative of f

sp4

A linear approximation of the function f involve

Vælg en eller flere:

☐

the Hessian of f

☒

the gradinet of f ✓

sp5

A quadratic approximation of the function f involve

Vælg en eller flere:

☒

the gradinet of f

☒

the Hessian of f ✓

Here's why:

The **quadratic (second-order Taylor) approximation** of a scalar function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ around a point \mathbf{x}_0 is:

$$f(\mathbf{x}) \approx f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^T (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2} (\mathbf{x} - \mathbf{x}_0)^T H_f(\mathbf{x}_0) (\mathbf{x} - \mathbf{x}_0)$$

Where:

- $\nabla f(\mathbf{x}_0)$: the **gradient** of f at \mathbf{x}_0 — this gives the **linear (first-order)** behavior.
- $H_f(\mathbf{x}_0)$: the **Hessian** of f at \mathbf{x}_0 — this gives the **curvature (second-order)** information.

So:

- ☐ **Gradient** is needed — tells us the slope (first-order).
- ☐ **Hessian** is needed — tells us how the slope changes (second-order).

In summary:

To build a **quadratic approximation**, you need:

- The **function value** at the point,
- The **gradient** (first derivatives),
- The **Hessian** (second derivatives).

sp6

A point $x' \in \mathbb{R}^n$ is a weak local minimizer of $f : \mathbb{R}^n \rightarrow \mathbb{R}$ if

Vælg en eller flere:

- ☐ $f(x) \geq f(x')$ for all x
- ☐ $f(x) \leq f(x')$ for all x sufficiently close to x'
- ☐ $f(x) \leq f(x')$ for all x
- ☒ $f(x) \geq f(x')$ for all x sufficiently close to x' ✓

sp7

A point $x' \in \mathbb{R}^n$ is a weak global minimizer of $f : \mathbb{R}^n \rightarrow \mathbb{R}$ if

Vælg en eller flere:

☒ $f(x) \geq f(x')$ for all x ✓

☐ $f(x) \leq f(x')$ for all x

☐ $f(x) \leq f(x')$ for all x sufficiently close to x'

☐ $f(x) \geq f(x')$ for all x sufficiently close to x'

sp8

If x' is a local minimizer of $f : \mathbb{R}^n \rightarrow \mathbb{R}$, with $f \in C^1$, then

Vælg en eller flere:

☒ the gradient of f at x' is zero ✓

☐ the gradient of f at x' is smaller than or equal to zero

☐ the gradient of f is zero

☐ the gradient of f at x' is bigger than or equal to zero

◆ Intuition:

- The gradient $\nabla f(x')$ points in the direction of steepest increase.
- If you're at a **local minimum**, there's **no direction** to go that would decrease the function further (at least locally).
- Therefore, the slope in **every direction** is zero — and this is exactly what the gradient being zero means.

sp9

If x' is a local minimizer of $f : \mathbb{R}^n \rightarrow \mathbb{R}$, with $f \in C^2$, then

Vælg en eller flere:

- ☐ the gradient of f at x' is zero or the hessian of f at x' is positive semi definite
- ☒ the gradient of f at x' is zero and the hessian of f at x' is positive semi definite ✓
- ☐ the gradient of f at x' is zero or the hessian of f at x' is positive definite
- ☐ the gradient of f at x' is zero and the hessian of f at x' is positive definite

sp10

A set $S \subseteq \mathbb{R}^n$ is convex if

Vælg en eller flere:

- ☒ $S = \{x \in \mathbb{R}^n \mid \|x\|_1 \leq 5\}$ ✓
- ☐ a curve between any two points in S is contained in S
- ☐ $S = \{x \in \mathbb{R}^n \mid \|x\|_2 = 1\}$ is a circle
- ☒ the straight line between any two points in S is contained in S ✓
- ☐ $n = 1$ and S is the union of two disjoint intervals
- ☒ $S = \{x \in \mathbb{R}^n \mid \|x\|_2 \leq 1\}$ is a disk ✓

✓ 1. $S = \{x \in \mathbb{R}^n \mid \|x\|_1 \leq 5\}$

- Yes — this set is **convex**.
- It's the **unit ball** in ℓ^1 -norm (scaled by 5).
- In **2D**, this looks like a **diamond shape** (not a box), with corners at $(5, 0), (0, 5), (-5, 0), (0, -5)$.
- In **higher dimensions**, it's a **cross-polytope**, not a box.
- ⚠️ **So, it's *not* a box, but it is convex.**

sp11

A function $f : R \rightarrow \mathbb{R}$, with $R \subseteq \mathbb{R}^n$ and $f \in C^2$, is convex if

Vælg en eller flere:

☒ f is a quadratic function ✓

☐ R is convex and the hessian $H(x)$ of f is positive semi definite for some $x \in R$

☒ R is convex and the hessian $H(x)$ of f is positive semi definite for all $x \in R$ ✓

☐ the hessian $H(x)$ of f is positive semi definite for all $x \in R$

☐ R is convex

☒ f is a linear function

✓ Correct answer 2:

" f is a linear function"

- Linear functions are of the form $f(x) = a^T x + b$.
- Linear functions are **both convex and concave**, because:

$$f(\lambda x + (1 - \lambda)y) = \lambda f(x) + (1 - \lambda)f(y)$$

- This is **stronger** than the convexity inequality, so linear functions are always convex.

sp12

Let $R \subseteq \mathbb{R}^n$. The optimization problem $\min_{x \in R} f(x)$ is convex if

Vælg en eller flere:

☒ R is convex and $f : R \rightarrow \mathbb{R}$ is convex ✓

☐ $f : S \rightarrow \mathbb{R}$ is convex for some (convex) set S with $R \subset S$

☐ R is convex

☐ R is convex and $f : S \rightarrow \mathbb{R}$ is convex for some (convex) set S with $S \subset R$

☒ $f : R \rightarrow \mathbb{R}$ is convex

☒ R is convex and $f : S \rightarrow \mathbb{R}$ is convex for some (convex) set S with $R \subset S$ ✓

✓ Correct Answer 2:

" $f : R \rightarrow \mathbb{R}$ is convex"

- This is **necessary**, but not **sufficient** on its own.
- However, it is still considered **part of what makes a problem convex**.
- If this option was presented alone, it might not be enough — but in the context of multiple correct answers, it's valid as a **partial requirement**.

✓ This is correct, assuming R is convex too.

✓ Correct Answer 3:

" R is convex and $f : S \rightarrow \mathbb{R}$ is convex for some (convex) set S with $R \subset S$ "

- This is also valid. If f is convex on a **larger convex set** S that contains R , and R is convex, then:
 - f is convex **on** R as well.
- Convexity is preserved when you **restrict a convex function to a convex subset** of its domain.

✓ So this also guarantees a convex optimization problem.

abc In plain language:

$f : R \rightarrow \mathbb{R}$ means:

- There is a **function** f ,
- It takes **inputs from the set** R ,
- And it **outputs real numbers** (i.e., values in \mathbb{R}).

sp13

Mark the statements which are true:

Vælg en eller flere:

- ☒ The software tool CVX is a numerical solver for optimization problems ✖
- ☐ Any optimization problem can be approximated arbitrary well with a convex optimization problem
- ☒ The solution set of a convex optimization problem is convex ✔
- ☐ Local minimizers are global minimizers in general
- ☒ For convex optimization problems local minimizers are global minimizers ✔
- ☐ Optimization problems are in general numerically solvable
- ☒ The software tool CVX is a numerical solver for convex optimization problems ✔
- ☒ Convex optimization problems are in general numerically solvable ✔

gradient methods

sp1

The steepest-descent method can be applied to

Vælg en eller flere:



least squares problems ✓



constrained optimization problems



unconstrained convex optimization problems ✓



unconstrained optimization problems

✓ 1. Unconstrained optimization problems

- **Steepest descent** is a method for **minimizing a function** without any constraints on the variables.
- It uses the **negative gradient direction** (steepest decrease) to iteratively update the variable x .
- So, any optimization problem without constraints is a valid candidate.

✦ Correct

✓ 2. Unconstrained convex optimization problems

- If the function is **convex** and **differentiable**, the **steepest descent method** is **guaranteed to converge** to a **global minimum**, assuming proper step sizes.
- Convexity makes the method **more reliable and efficient**.

✦ Correct and even better than the general case

sp2

The steepest-descent method is a

Vælg en eller flere:



first order method ✓



third order method



fourth order method



second order method

sp3

Let $f \in C^2$ be a cost function, H its hessian, and $\{x_k\}$ denote the sequence generated by the steepest-descent algorithm. For x_k close to a minimizer x^* the algorithm converge fast to x^* if

Vælg en eller flere:



the eigenvalues of $H(x_k)$ are almost the same



the eigenvalues of $H(x_k)$ are small



all eigenvalues of $H(x_k)$ are real



the largest and smallest eigenvalue of $H(x_k)$ are almost the same ✓

(condition Number)

sp4

The Steepest-Descent Algorithm (SDA) and the Newton-Raphson Algorithm (NRA) can be used in combination by

Vælg en eller flere:



using the NRA fare from the minimizer and the SDA close to the minimizer



using the SDA fare from the minimizer and the NRA close to the minimizer ✓



iterating between the SDA and the NRA

sp5

The Newton-Raphson method is a

Vælg en eller flere:

☒ second order method ✓

☐ fourth order method

☐ third order method

☐ first order method

sp6

When attempting to solve $0 = \nabla f(x)$ for x , one can apply

Vælg en eller flere:

☐ a first order method

☒ the Gauss-Newton method ✓

☐ the steepest-descent method

☐ the Newton-Raphson method

🔍 What does $0 = \nabla f(x)$ mean?

You're looking for a point x where the **gradient of f** is zero — in other words, a **stationary point**, which could be a minimum, maximum, or saddle point.



Final Summary:

Reason Gauss-Newton is good	Explanation
Designed for least squares	Solves problems like $\min \ r(x)\ ^2$
Solves $\nabla f(x) = 0$	Because $\nabla f(x) = J^T r$
Uses Jacobians only	No need for second derivatives (Hessian)
Faster than Newton	Approximates Hessian with $J^T J$
Converges well	Especially when residuals are small

constrained optimization

PLEASE WRITE WHY?!?!?!?!?!?

sp1

Consider the constrained optimization problem $\min f(x) \quad s.t. \quad \mathbf{a}(x) = (a_1(x), \dots, a_p(x)) = 0$, and let $J(x) = J_{\mathbf{a}}(x)$ denote the Jacobian of \mathbf{a} at x . The point x' is regular if

Vælg en eller flere:

- ☒ $\mathbf{a}(x') = 0$ and the gradients $\nabla a_1(x'), \dots, \nabla a_p(x')$ are linear independent
- ☒ $a_i(x') = 0$ for $i = 1, \dots, p$ and $J(x)$ has full row rank
- ☐ $\text{Rank}(A) = p$ in the case where $\mathbf{a}(x) = Ax - b$, $A \in \mathbb{R}^{p \times n}$
- ☐ $\mathbf{a}(x') = 0$ and some of the gradients $\nabla a_1(x'), \dots, \nabla a_p(x')$ are linear independent
- ☒ $\mathbf{a}(x') = 0$ and $J(x')$ has full row rank
- ☒ $J(x')$ has full row rank
- ☒ the gradients $\nabla a_1(x'), \dots, \nabla a_p(x')$ are linear independent
- ☒ $Ax' = b$ and $\text{Rank}(A) = p$ in the case where $\mathbf{a}(x) = Ax - b$, $A \in \mathbb{R}^{p \times n}$
- ☐ $a_i(x') = 0$ for some i and $J(x)$ has full row rank

sp2

Consider the constrained optimization problem $\min f(x) \quad s.t. \quad \mathbf{a}(x) = (a_1(x), \dots, a_p(x)) = 0$, $\mathbf{c}(x) = (c_1(x), \dots, c_q(x)) \geq_e 0$, and let $J_{\mathbf{a}}(x)$, $J_{\mathbf{c}}(x)$ denote the Jacobian of \mathbf{a} , \mathbf{c} at x . The point x' is regular if

Vælg en eller flere:

- ☒ $\mathbf{a}(x') = 0$, $\mathbf{c}(x') \geq_e 0$ and the gradients $\nabla a_1(x'), \dots, \nabla a_p(x'), \nabla c_{j_1}(x'), \dots, \nabla c_{j_k}(x')$ are linearly independent, with $c_{j_l}(x') = 0$ for $l = 1, \dots, k$
- ☐ $\mathbf{a}(x') = 0$, $\mathbf{c}(x') \geq_e 0$ and the gradients $\nabla a_1(x'), \dots, \nabla a_p(x'), \nabla c_{j_1}(x'), \dots, \nabla c_{j_k}(x')$ are linearly independent, with $c_{j_l}(x') > 0$ for $l = 1, \dots, k$
- ☒ $\mathbf{a}(x') = 0$, $\mathbf{c}(x') \geq_e 0$ and the gradients $\nabla a_1(x'), \dots, \nabla a_p(x')$ are linearly independent
- ☐ $\mathbf{a}(x') = 0$, $\mathbf{c}(x') \geq_e 0$ and the gradients $\nabla a_1(x'), \dots, \nabla a_p(x'), \nabla c_{j_1}(x'), \dots, \nabla c_{j_k}(x')$ are linearly independent, with $c_{j_l}(x') \geq 0$ for $l = 1, \dots, k$
- ☒ $\mathbf{a}(x') = 0$, $\mathbf{c}(x') \geq_e 0$ and $J_{\mathbf{a}}(x')$ has full row rank
- ☐ $a_i(x') = 0$ for $i = 1, \dots, p$, $c_i(x') \geq 0$ for $i = 1, \dots, q$ and $J_{\mathbf{a}}(x)$ has full row rank

◆ 2. What is a **regular point**?

As we discussed earlier, a point x' is **regular** if the gradients of the **active constraints** (equalities and active inequalities) are **linearly independent**.

☐ **So, the answer is:**

A regular point is defined **with respect to the active constraints**.

- **Yes, it involves active constraints** — because **only active constraints** matter when checking **regularity** (constraint qualification).
- You do **not** require **all** constraints to be active — **only the active ones** must have linearly independent gradients.

sp3

Consider the constrained optimization problem $\min f(x) \quad s.t. \quad \mathbf{a}(x) = (a_1(x), \dots, a_p(x)) = 0$, and let $J(x') = J_{\mathbf{a}}(x')$ denote the Jacobian of \mathbf{a} at x' . Let the point x' be a regular minimizer (with $\nabla f(x') \neq 0$)

Vælg en eller flere:

- ☒ then $\nabla f(x')$ is in the range of $J(x')$
- ☐ then $\nabla f(x')$ is orthogonal to the gradients $\nabla a_1(x'), \dots, \nabla a_p(x')$
- ☒ then $\nabla f(x')$ is in the range of $J(x')^T$
- ☒ then $\nabla f(x')$, $\nabla a_1(x'), \dots, \nabla a_p(x')$ are linearly independent
- ☒ then $\nabla f(x')$ can be expressed as a linear combination of the gradients $\nabla a_1(x'), \dots, \nabla a_p(x')$
- ☒ then $\nabla f(x')$ is in the image of $J(x')^T$
- ☐ then $\nabla f(x')$ is in the image of $J(x')$

Lagrange multipliers and the Jacobian

To handle the constraint $a(x) = 0$, we define the **Lagrangian**:

$$\mathcal{L}(x, \lambda) = f(x) + \lambda^\top a(x)$$

To find critical points (candidates for minima), we compute:

$$\nabla_x \mathcal{L}(x, \lambda) = \nabla f(x) + J(x)^\top \lambda = 0$$

So we get the **first-order optimality condition**:

$$\nabla f(x) = -J(x)^\top \lambda$$

Why transpose?

Because:

- $J(x) \in \mathbb{R}^{p \times n}$
- $\lambda \in \mathbb{R}^p$
- So $J(x)^\top \lambda \in \mathbb{R}^n$, which **matches the dimension** of $\nabla f(x) \in \mathbb{R}^n$



That's how you can express the gradient of the objective function as a **linear combination** of the gradients of the constraints:



sp4

Consider the constrained optimization problem $\min f(x) \quad \text{s.t.} \quad a(x) = (a_1(x), \dots, a_p(x)) = 0, \quad c(x) = (c_1(x), \dots, c_q(x)) \geq_e 0$. Let the point x' be a regular minimizer (with $\nabla f(x') \neq 0$)

Vælg en eller flere:

- ☒ then $\nabla f(x')$ is a linear combination of $\nabla a_1(x'), \dots, \nabla a_p(x'), \nabla c_{j_1}(x'), \dots, \nabla c_{j_k}(x')$ with $c_{j_l}(x') = 0$ for $l = 1, \dots, k$ 
-  then $\nabla f(x')$ is a linear combination of $\nabla a_1(x'), \dots, \nabla a_p(x'), \nabla c_{j_1}(x'), \dots, \nabla c_{j_k}(x')$
- ☐ then $\nabla f(x')$ is a linear combination of $\nabla a_1(x'), \dots, \nabla a_p(x'), \nabla c_{j_1}(x'), \dots, \nabla c_{j_k}(x')$ with $c_{j_l}(x') > 0$ for $l = 1, \dots, k$
- ☐ then $\nabla f(x')$ is a linear combination of $\nabla a_1(x'), \dots, \nabla a_p(x'), \nabla c_{j_1}(x'), \dots, \nabla c_{j_k}(x')$ with $c_{j_l}(x') \geq 0$ for $l = 1, \dots, k$

Til svar 2

✓ Short answer:

Yes, it's **technically true, but misleading** — and **not useful** in practice — because:

- Only the **gradients of the active constraints** (i.e., constraints that are **binding** at the point x') will have **non-zero multipliers**.
- Gradients of **inactive inequality constraints** (those where $c_j(x') > 0$) will appear with multiplier $\mu_j = 0$.

So while you can **write** the linear combination as involving all $\nabla c_j(x')$, the inactive ones just have a **zero coefficient**.

sp5

Consider the constrained optimization problem $\min f(x) \quad s. t. \quad \mathbf{a}(x) = (a_1(x), \dots, a_p(x)) = 0, \mathbf{c}(x) = (c_1(x), \dots, c_q(x)) \geq_e 0$. Let the point x' be a regular minimizer and λ', μ' be the corresponding multipliers

Vælg en eller flere:

- ☐ If $\mu_i > 0$ then $c_i(x') > 0$
- ☒ If $\mu_i > 0$ then $c_i(x') = 0$
- ☐ If $\mu_i > 0$ then $c_i(x') < 0$
- ☒ If $c_i(x') > 0$ then $\mu_i = 0$
- ☐ If $\nabla f(x') = 0$ then $\lambda' \neq 0, \mu' = 0$
- ☐ If $c_i(x') > 0$ then $\mu_i \neq 0$
- ☐ If $\nabla f(x') = 0$ then $\lambda' = 0, \mu' > 0$
- ☒ If $\nabla f(x') = 0$ then $\lambda' = 0, \mu' = 0$

sp6

Consider the constrained optimization problem

$\min f(x) \quad s.t \quad \mathbf{a}(x) = (a_1(x), \dots, a_p(x)) = 0, \mathbf{c}(x) = (c_1(x), \dots, c_q(x)) \geq_e 0$. The KKT conditions are necessary and sufficient when

Vælg en eller flere:

☒ the constrained optimization problem is convex ✓

☐ $-f$ and $-\mathbf{c}$ are convex and \mathbf{a} is affine linear

☒ f and $-\mathbf{c}$ are convex and \mathbf{a} is affine linear

☒ f, \mathbf{a} and $-\mathbf{c}$ are convex ✗

☐ f and \mathbf{c} are convex and \mathbf{a} is affine linear

☒ the constrained optimization problem is a linear programming problem

☹ Your answer is partially correct.

Du har 1 rigtig(e).

f and $-\mathbf{c}$ are convex and \mathbf{a} is affine linear

De rigtige svar er:

**, the constrained optimization problem is convex,
the constrained optimization problem is a linear programming problem**

Affine linear, means linear function where affine means with shift, there an affine linear is a straight line not passing through origin, it needs to have a shift

intro ml and bayesian theory

sp1

Which of the following is an example of unsupervised learning?

☒ a. Finding groups among customers based on their purchasing behavior.

☐ b. Predicting house prices using historical sales data.

☐ c. Classifying emails as spam or not spam using historical data.

☐ d. None of the above. ✗

☐ a. Finding groups among customers based on their purchasing behavior

This is an example of **unsupervised learning** because:

- You are not given any labeled data (i.e., you're not told in advance what the "groups" should be).
- The algorithm is expected to **discover patterns or groupings (clusters)** in the data on its own.
- A common technique for this is **clustering** (e.g., k-means clustering).

Now, let's explain **why the other options are incorrect** (i.e., they are not examples of unsupervised learning):

☐ **b. Predicting house prices using historical sales data**

- This is an example of **supervised learning** because:
- The algorithm is trained on **input-output pairs** (features like square footage, location → house price).
- The goal is to **predict a specific label or value** (the house price).
- This is a **regression task**.

☐ **c. Classifying emails as spam or not spam using historical data**

- Also **supervised learning** because:
- You have **labeled data** (emails labeled as "spam" or "not spam").
- The algorithm learns from these examples to classify new emails.
- This is a **classification task**.

☐ **d. None of the above**

- This would be correct only if **none** of the options described unsupervised learning, but **option a** clearly does.

sp2

A model that perfectly fits the training data but performs poorly on unseen data is most likely experiencing:

☒ a. Low bias and high variance

☐ b. High bias and low variance *

☐ c. High bias and high variance

☐ d. Low bias and low variance

💡 The Key Idea: Overfitting

A model that **perfectly fits the training data** but **performs poorly on unseen (test) data** is **overfitting**. What happens in overfitting?

- The model **memorizes** the training data instead of learning general patterns.
- It performs **very well on training data** but **poorly on test data**.

- This is a sign of:
- **Low bias** (because it fits the training data almost perfectly — not underestimating the complexity).
- **High variance** (because it's too sensitive to fluctuations/noise in the training data — generalizes poorly).

sp3

Which of the following is true for random variables X and Y ?

- ☐ a. None of the statements is true
- ☐ b. $p_{XY}(x, y) = p_{Y|X}(x | y)p(X)/p(Y)$
- ☐ c. $p_{X,Y}(x, y) = p_X(x) + p_Y(y) - p_X(x)p_Y(y)$
- ☒ d. $p_{X,Y}(x, y) = p_X(x)p_Y(y)$ ✓

🔍 Why is this correct?

Joint Probability for Independent Variables:

If two random variables X and Y are **independent**, their **joint probability distribution** is:

$$p_{X,Y}(x, y) = p_X(x) \cdot p_Y(y)$$

This means the probability of both events happening together is the product of their individual probabilities.

sp4

Under a 0-1 loss function in Bayesian decision theory, the optimal classification decision is determined by:

- ☒ a. C) Maximizing the posterior probability of the class ✓
- ☐ b. Minimizing the prior probability of misclassification
- ☐ c. Maximizing the likelihood of the observed data
- ☐ d. Minimizing the conditional risk for all possible actions

slideee 39 and 40 from lecture intro to ML and bayesiaa theory

sp5

For the rejection option a_{reject} , we assign a loss $L(y, a_{\text{reject}}) = \lambda$ with $\lambda \in [0, 1]$. What will the expected risk of taking the rejection option?

- ☐ a. $1/\lambda$
- ☐ b. It depends on the prior of classes $p_Y(y)$.
- ☒ c. $1 - \lambda$ ✗
- ☐ d. λ

slide 42 from lecture intro to ML and bayesiaa theory

parametric and nonparametric

sp1

What is the key difference between parametric and nonparametric methods?

- ☐ a. Nonparametric methods have a fixed number of parameters.
- ☐ b. Nonparametric methods cannot perform classification.
- ☒ c. Parametric methods use a fixed set of parameters, while nonparametric methods grow in complexity with data. ✓
- ☐ d. Parametric methods always use Gaussian distributions.

sp2

If we have N independent and identically distributed measurements X_1, X_2, \dots, X_N from the Exponential distribution with parameter λ ($p_X(x) = \lambda e^{-\lambda x}$), the log-likelihood will be
 $\ell(\lambda) = \sum_{i=1}^N \log(\lambda) - \lambda x_i$
 The maximum likelihood estimator $\hat{\lambda}_{MLE}$ is

- ☐ a. $\hat{\lambda}_{MLE} = e^{\frac{1}{N} \sum_{i=1}^N x_i}$
- ☐ b. $\hat{\lambda}_{MLE} = \frac{1}{N} \sum_{i=1}^N x_i$
- ☒ c. $\hat{\lambda}_{MLE} = \frac{N}{\sum_{i=1}^N x_i}$
- ☐ d. $\hat{\lambda}_{MLE} = \log(\frac{1}{N} \sum_{i=1}^N x_i)$ ✖

sp3

Our data contains three classes with two continuous features. We have known prior probabilities for each class. If we use Gaussian naive Bayes as a classifier, how many parameters do we have to estimate?

- ☐ a. 15
- ☒ b. 12 ✔
- ☐ c. 9
- ☐ d. 6

1 2 3 4 Let's do the math:

- 3 classes
- 2 features
- 2 parameters per feature per class (mean and variance)

So the total number of parameters:

$$3 \text{ classes} \times 2 \text{ features} \times 2 \text{ parameters} = \boxed{12}$$

sp4

One of the following is the posterior distribution if we used K nearest neighbor to model the evidence and the likelihood in a classification task

- ☐ a. None of the above
- ☐ b. $\frac{N_c \text{ (counts of class c)}}{k \text{ (number of neighbors)}}$
- ☐ c. $\frac{k_c \text{ (counts of neighbors from class c)}}{N \text{ (Total number of data)}}$
- ☒ d. $\frac{k_c \text{ (counts of neighbors from class c)}}{k \text{ (number of neighbors)}}$ ✔

💡 What does this represent?

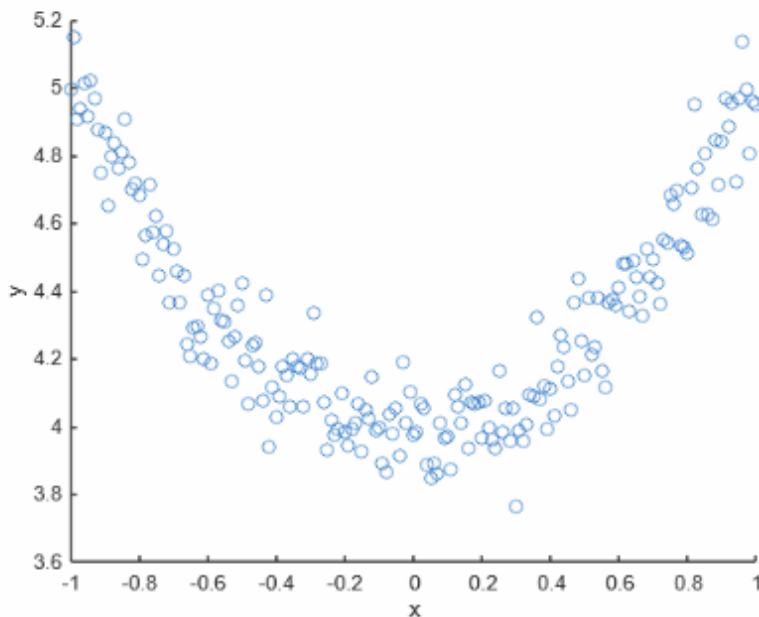
In K-nearest neighbors (K-NN) classification, we:

- Look at the **k nearest neighbors** of a query point (based on some distance metric).
- Count **how many of those neighbors belong to each class**.
- Predict the class with the highest count (i.e., majority vote).

linear regression

sp1

For the following data:



Which of the following basis functions would be the most appropriate for a linear regression model?

☐ a. $\phi_0(x) = x, \phi_1(x) = x^2$ ✗

☒ b. $\phi_0(x) = 1, \phi_1(x) = x^2$

☐ c. $\phi_0(x) = 1, \phi_1(x) = \exp(x)$

☐ d. $\phi_0(x) = \sin(x), \phi_1(x) = \cos(x)$

□ Correct Answer:

b. $\phi_0(x) = 1, \phi_1(x) = x^2$

Why it's **correct**:

- $\phi_0(x) = 1$: This gives you the **intercept (bias)** term in the linear model.
- $\phi_1(x) = x^2$: This captures the **quadratic curvature** needed to fit the U-shaped pattern.

So the linear regression model would be:

$$y \approx \beta_0 \cdot 1 + \beta_1 \cdot x^2 = \beta_0 + \beta_1 x^2$$

This matches the shape of your data.

✗ Your Chosen Answer:

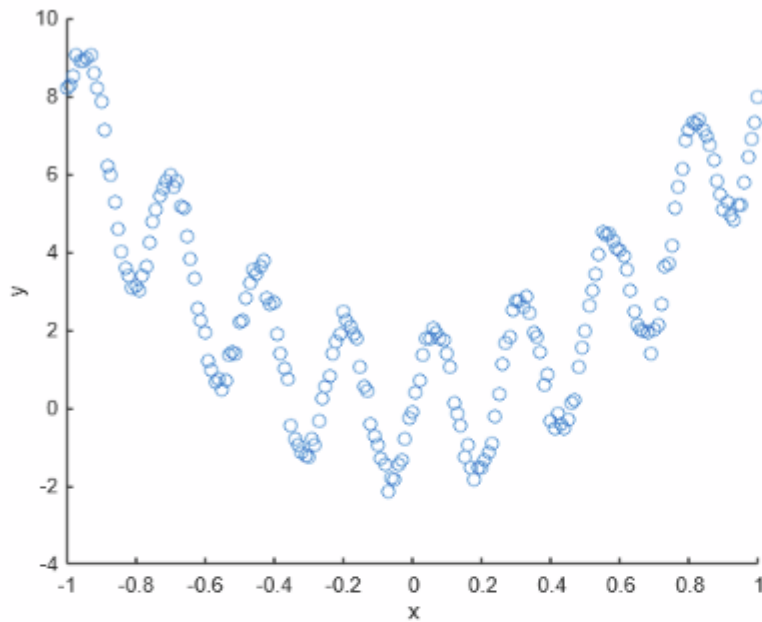
a. $\phi_0(x) = x, \phi_1(x) = x^2$

Why it's **wrong**:

- This doesn't include a **constant (bias) term**.
- Without a bias (intercept), the model **forces the curve to pass through the origin**, which is not appropriate here. Your data has a clear vertical shift — it's centered around $y \approx 4$, not 0.

sp2

For the following data:



Which of the following basis functions would be the most appropriate for a linear regression model?

- ☐ a. $\phi_0(x) = 1, \phi_1(x) = \sin(2\pi fx)$
- ☐ b. $\phi_0(x) = \cos(2\pi fx), \phi_1(x) = \sin(2\pi fx)$
- ☐ c. $\phi_0(x) = \cos(2\pi fx), \phi_1(x) = x$
- ☒ d. $\phi_0(x) = x^2, \phi_1(x) = \sin(2\pi fx)$ ✓

sp3

In linear regression with a Gaussian likelihood, which of the following prior distributions on the weights will result in a maximum a posteriori estimate \hat{w}_{MAP} which is equivalent to having a regularization on the weights being $\|w\|_1$?

- ☐ a. Exponential Distribution
- ☒ b. Laplace distribution ✓
- ☐ c. Beta distribution
- ☐ d. Gaussian Distribution

sp4

In linear regression with a Gaussian likelihood, if we assign the following prior on the weights $p_W(w) \propto \exp(-\|w\|^4)$, then the maximum a posteriori solution will be the solution of the least squares with which of the following regularizations?

☐ a. None of the mentioned options.

☒ b. $R(w) = \|w\|^2$ ✖

☐ c. $R(w) = \exp(\|w\|^4)$

☒ d. $R(w) = \|w\|^4$

1. Prior given:

$$p(w) \propto \exp(-\|w\|^4)$$

This is **not** a standard Gaussian prior (which would give ridge/L2 regularization). It's a **heavier-tailed prior** that penalizes large weights more than L2 but less than L1 near zero.

2. MAP = maximize posterior = maximize log posterior

MAP estimation becomes:

$$\hat{w}_{MAP} = \arg \max_w [\log p(\text{data} | w) + \log p(w)]$$

Substitute the prior:

$$\log p(w) \propto -\|w\|^4$$

So the optimization becomes:

$$\hat{w}_{MAP} = \arg \min_w [\text{Least Squares Loss} + \lambda \cdot \|w\|^4]$$

This is equivalent to **least squares with an $\|w\|^4$ regularization term**.

linear classification

sp1

Why do we need to modify the linear regression model for linear classification?

- ☐ a. Because linear regression is sensitive to outliers. ❌
- ☒ b. Because linear regression predicts continuous values, but classification requires discrete class labels.
- ☐ c. Because linear regression cannot handle high-dimensional data.
- ☐ d. Because linear regression always overfits the data.

Here's why this is correct:

Linear regression is designed to **predict continuous numeric values**. For example, it might predict a house price like 243,500 dollars based on input features.

However, in **classification tasks**, we are trying to predict **discrete labels** — for example, whether an email is "spam" or "not spam", or whether a tumor is "malignant" or "benign". These are **categories**, not continuous values.

If we try to use linear regression for classification, it will output a continuous number (like 0.8, -1.3, or 2.5), which doesn't directly tell us which class the example belongs to.

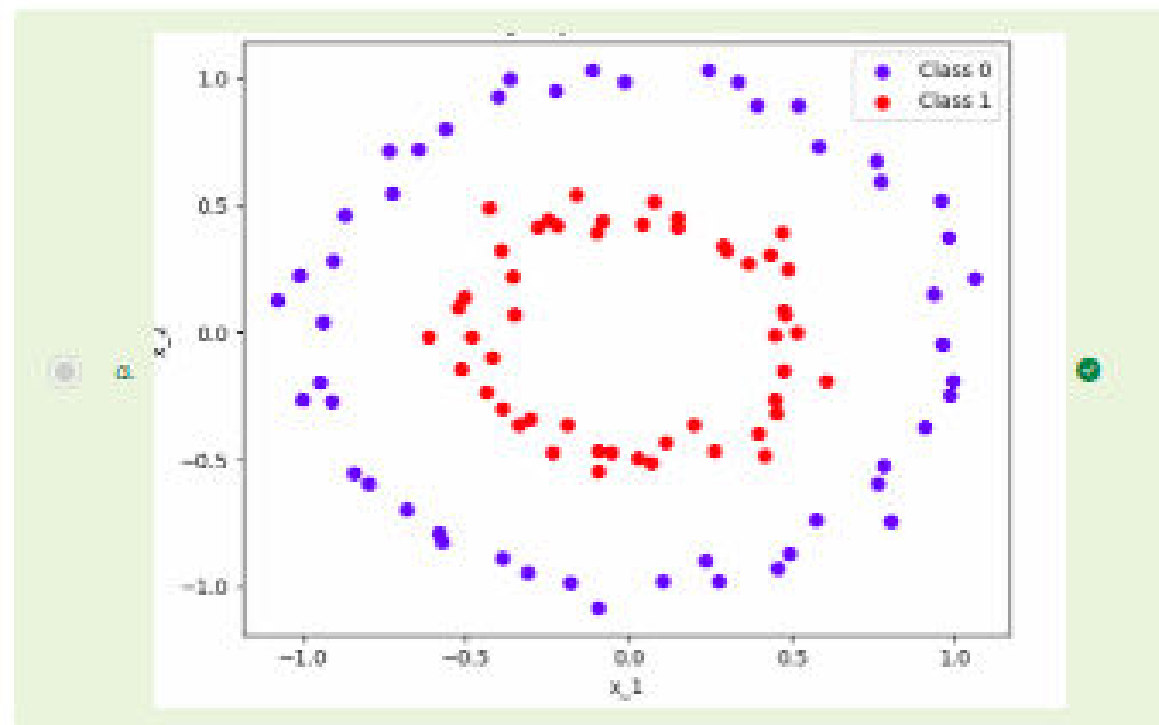
To make a classification decision, we need to **transform the output** into a class label — for instance, by applying a threshold (e.g., if the output is > 0.5 , classify as 1, otherwise 0). But this is not how linear regression is intended to work, and it doesn't model the **probability** of class membership properly.

Instead, for classification, we use models like **logistic regression**, which are specifically designed to:

- Output values between 0 and 1 (interpretable as probabilities)
- Model the relationship between input features and **discrete class labels**

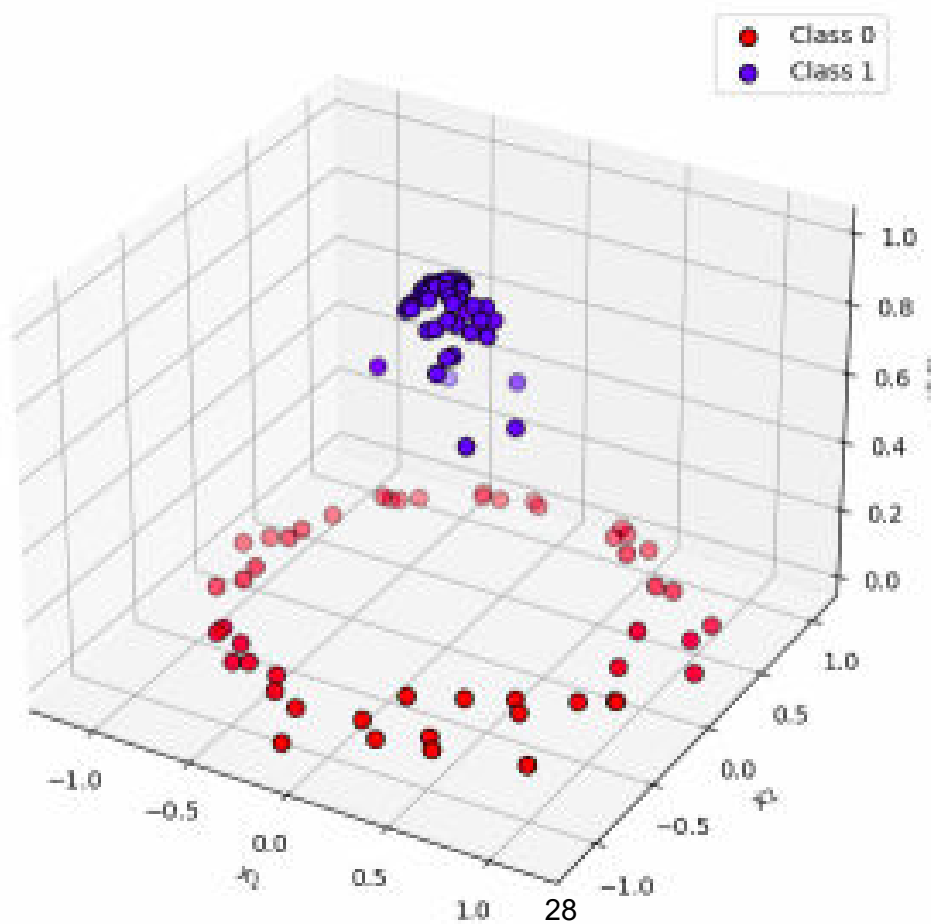
sp2

For which of the following datasets will the perceptron algorithm fail to converge?



- ☐ b. It will converge for both of the datasets.
- ☐ c. It will not converge for both of the datasets.

☐ d.



has to be 3 dimension to separate data

sp3

In logistic regression for binary classification, what is the role of the logistic sigmoid function

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

- ☐ a. It guarantees that the decision boundary is non-linear.
- ☐ b. It maps the output to a binary decision of 0 or 1.
- ☒ c. It provides a smooth, differentiable function that outputs probabilities between 0 and 1. ✓
- ☐ d. It scales the input features to have a zero mean.

sp4

The error in the perceptron algorithm is

- ☐ a. $E^p(w) = \sum_{n=1}^N -w^\top \phi(x_n) y_n$
- ☐ b. $E^p(w) = \sum_{n=1}^N \|y_n - w^\top \phi(x_n)\|_2^2$

- ☒ c. $E^p(w) = \sum_{n \in \mathcal{M}} w^\top \phi(x_n) y_n$
where \mathcal{M} is the set of misclassified training data. ✗

- ☒ d. $E^p(w) = \sum_{n \in \mathcal{M}} -w^\top \phi(x_n) y_n$
where \mathcal{M} is the set of misclassified training data.

✓ What is the **perceptron error function**?

The perceptron only "cares" about the examples it misclassifies. For a **misclassified point**, the condition is:

$$y_n w^\top \phi(x_n) \leq 0$$

So, for those misclassified points, the **perceptron error function** is defined as:

$$E_p(w) = \sum_{n \in M} -y_n w^\top \phi(x_n)$$

Here's why:

- If $y_n w^\top \phi(x_n) > 0$, the point is correctly classified → **not included in the error**.
- If $y_n w^\top \phi(x_n) \leq 0$, it's misclassified → **include in the error**.
- The negative sign (-) ensures that **misclassified points contribute positively to the error**, so minimizing the error encourages **correct classification**.

✓ So to summarize:

- The **perceptron error function** adds up contributions **only from misclassified points**.
- It uses $-y_n w^\top \phi(x_n)$ so that misclassified points add **positive error**.
- This is why the correct answer is:

$$E_p(w) = \sum_{n \in M} -y_n w^\top \phi(x_n)$$

intro to NN

sp1

Given inputs x_1, \dots, x_d with $x_0 = 1$, which of the following formulas describes, **in general**, how the neuron or the general perceptron in a multilayer perceptron computes its activation z ?

- ☒ a. $z = h(a)$ with $a = \sum_{m=0}^d w_m x_m$ and $h(\cdot)$ being an activation function. ✓
- ☐ b. $z = h(a)$ with $a = \sum_{m=1}^d w_m x_m$ and $h(\cdot)$ being an activation function.
- ☐ c. $z = h(a)$ with $a = \prod_{m=0}^d w_m x_m$ and $h(\cdot)$ being an activation function.
- ☐ d. $z = h(a)$ with $a = \sum_{m=0}^d w_m h(x_m)$ and $h(\cdot)$ being an activation function.

sp2

Which of the following is true?

- ☐ a. XOR can be represented by a perceptron.
- ☐ b. NAND (Not AND) cannot be represented by a perceptron.
- ☐ c. Multilayer perceptrons can only represent linearly separable data.
- ☒ d. None of these expressions is true ✓

A single-layer perceptron can **only solve linearly separable problems**

Multilayer perceptrons (MLPs) with one or more hidden layers are **universal function approximators**.

They can represent **non-linearly separable data** (including XOR).

sp3

Which of the following is true?

- ☐ a. The function $f(x) = \cos(x)$ defined on $x \in [0, 1]$ cannot be represented by an MLP.
- ☐ b. The function $f(x) = \cos(x)$ defined on $x \in [0, 1]$ needs at least two hidden layers to be represented by an MLP.
- ☐ c. All of the statements are wrong.
- ☒ d. The function $f(x) = \cos(x)$ defined on $x \in [0, 1]$ needs at least one hidden layer to be represented by an MLP. ✓

Key insight:

One hidden layer in an MLP (with enough neurons) is enough to approximate any continuous function on a compact domain, including sums of nonlinear functions.

Why?



- The **Universal Approximation Theorem** says that a **single hidden layer neural network with nonlinear activation functions can approximate any continuous function** on a closed and bounded interval, **to any desired accuracy, as long as you have enough neurons in that layer.**
- This applies to functions like sums of cosines, sines, polynomials, or any other continuous nonlinear functions combined.

When do you need more hidden layers?

- **Depth can help efficiency:**
While one hidden layer can approximate the function, **deeper networks** (more hidden layers) often can do it **more efficiently** — with fewer neurons or better generalization.
- **Some complex functions are easier to approximate with deeper networks**, especially when the function has a hierarchical or compositional structure.
- For example, if your function is a **composition** of nonlinear functions (like $f(x) = g(h(x))$), deeper networks shine.

sp4

We want a neural network to predict the coffee-to-water ratio based on the brewing method, coffee choice, and the customer's taste preference. Which of the following activation functions for the output layer is the most suitable for this application?

- ☐ a. Sign
- ☐ b. Identity
- ☒ c. ReLU 
-  d. Sigmoid

◆ So what kind of output activation function do we need?

We need one that:

- Outputs a **real-valued scalar**
- Ideally **bounded**, if we know the ratio should stay in a certain range (like 0–1)
- Allows for **smooth, continuous output**

Correct answer: Sigmoid

- The coffee-to-water ratio is a **real number between 0 and 1**.
- **Sigmoid** outputs values in **(0, 1)** — perfect for this case.
- Other options:
- **Sign**: only -1 or 1 → \square
- **Identity**: unbounded output → \triangle could go below 0 or above 1
- **ReLU**: no upper limit → \triangle might produce too-large values

sp5

What is the primary purpose of the backpropagation algorithm in training neural networks?

- ☐ a. To adjust the network's weights randomly until performance improves.
- ☐ b. To calculate the activations for each neuron during the forward pass.
- ☐ c. To determine the optimal activation functions for each neuron.
- ☒ d. To compute the gradient of the loss function with respect to each weight via the chain rule. ✓