UCLA-ALIGNED COURSE FIELD NOTES

STATS 102B

Introduction to Computation and Optimization for Statistics

Matrix computation, dimension reduction, clustering, numerical optimization, EM, and dynamic programming

How these notes are written. I wrote each chapter the way I wish the material had been explained the first time: the real idea, the point where I usually get stuck, the simplest mental model that still stays honest, and a worked decision instead of a polished answer appearing from nowhere. Every chapter closes with practice and a fully written solution. If a sentence sounds impressive but does not help me solve or explain something, it does not belong here.

Daniel Mastick

Curriculum map, working notes, practice, and solutions

Course map

Week	Core thread	What should be solid by the end
1	**Numerical linear algebra and stable computation**	Reconstruct the model, work an application, test its assumptions, and explain the result in ordinary language.
2	**Multivariate normal computation**	Reconstruct the model, work an application, test its assumptions, and explain the result in ordinary language.
3	**Principal components and singular value decomposition**	Reconstruct the model, work an application, test its assumptions, and explain the result in ordinary language.
4	**Clustering objectives and algorithms**	Reconstruct the model, work an application, test its assumptions, and explain the result in ordinary language.
5	**Gradient, Newton, and constrained optimization**	Reconstruct the model, work an application, test its assumptions, and explain the result in ordinary language.
6	**Stochastic optimization and statistical loss**	Reconstruct the model, work an application, test its assumptions, and explain the result in ordinary language.
7	**Expectation-maximization and latent variables**	Reconstruct the model, work an application, test its assumptions, and explain the result in ordinary language.
8	**Dynamic programming and reproducible algorithm design**	Reconstruct the model, work an application, test its assumptions, and explain the result in ordinary language.
9	**Integration studio**	Combine several chapters in one case, preserve their assumptions, and reconcile the result across representations.
10	**Cumulative review**	Retrieve the full course map from memory and solve mixed problems without chapter labels.

Scope anchor: https://catalog.registrar.ucla.edu/course/2025/STATS102B. The sequence is aligned to the official catalog description and expanded into a practical ten-week study plan.

How I would use this packet

Treat each numbered section as one chapter. Read the formal explanation, pause at the student interpretation, and see whether the easy version still says the same thing. Rework the example without looking, then cover the solution in the chapter practice box. The cumulative set at the end is deliberately mixed: knowledge becomes durable when I have to choose the tool instead of being told which chapter I am in.

Contents

CHAPTER 1 Numerical linear algebra and stable computation

The idea

Statistical computing relies on matrix products, factorizations, linear solves, eigenproblems, and singular values. Conditioning and floating-point error explain why solving a system is preferable to explicitly forming an inverse.

Rebuild the chapter from first principles

The claim I need to own is this: Statistical computing relies on matrix products, factorizations, linear solves, eigenproblems, and singular values. Conditioning and floating-point error explain why solving a system is preferable to explicitly forming an inverse. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of numerical linear algebra and stable computation. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach numerical linear algebra and stable computation on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from the course foundations to multivariate normal computation. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

The full working model

- **Core claim.** Statistical computing relies on matrix products, factorizations, linear solves, eigenproblems, and singular values.
- **What moves.** Conditioning and floating-point error explain why solving a system is preferable to explicitly forming an inverse.
- **What keeps the answer honest.** The model becomes useful only when I can apply numerical linear algebra and stable computation to an unfamiliar case and defend the assumptions.
- **Mastery standard.** I can define the objects, rebuild the rule, work a fresh case, compare alternatives, and diagnose a broken solution without relying on recognition.

How I actually think about it

I treat algebraically equivalent formulas as computationally different algorithms.

The easy version

Statistical computing relies on matrix products, factorizations, linear solves, eigenproblems, and singular values.

Worked example

Question. Why solve X beta=y instead of computing inverse(X)y?
Walkthrough. A factorization-based solve is faster and usually more numerically stable. Explicit inversion adds rounding error and unnecessary work.

Common miss

The fastest way to get this chapter wrong is to use the visible procedure without naming its assumption, units, time period, reference group, or boundary. I put that condition beside the work and verify the result independently.

Solved chapter practice

Practice 1. Why solve X beta=y instead of computing inverse(X)y?
Solution. A factorization-based solve is faster and usually more numerically stable. Explicit inversion adds rounding error and unnecessary work.

Practice 2. Explain numerical linear algebra and stable computation to a classmate using one definition, one concrete example, one assumption, and one failure mode.
Solution. Begin with the core claim above, use the worked case as the concrete example, state the condition highlighted by the warning, and choose a failure where that condition does not hold. A complete explanation connects all four pieces instead of listing them.

Mastery practice A. Reconstruct numerical linear algebra and stable computation from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a numerical linear algebra and stable computation problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 2 Multivariate normal computation

The idea

The multivariate normal is governed by a mean vector and covariance matrix. Quadratic forms, Cholesky factors, conditional distributions, simulation, and log-determinants support likelihoods without unstable determinant or inverse calculations.

Rebuild the chapter from first principles

The claim I need to own is this: The multivariate normal is governed by a mean vector and covariance matrix. Quadratic forms, Cholesky factors, conditional distributions, simulation, and log-determinants support likelihoods without unstable determinant or inverse calculations. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of multivariate normal computation. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach multivariate normal computation on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from numerical linear algebra and stable computation to principal components and singular value decomposition. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

The full working model

- **Core claim.** The multivariate normal is governed by a mean vector and covariance matrix.
- **What moves.** Quadratic forms, Cholesky factors, conditional distributions, simulation, and log-determinants support likelihoods without unstable determinant or inverse calculations.

- **What keeps the answer honest.** The model becomes useful only when I can apply multivariate normal computation to an unfamiliar case and defend the assumptions.
- **Mastery standard.** I can define the objects, rebuild the rule, work a fresh case, compare alternatives, and diagnose a broken solution without relying on recognition.

How I actually think about it

I read covariance matrices geometrically as scale and orientation, then use factorizations to compute with them.

The easy version

The multivariate normal is governed by a mean vector and covariance matrix.

Worked example

Question. How can Z from a standard multivariate normal produce covariance Sigma?
Walkthrough. Find a Cholesky factor L with LL-transpose=Sigma and set X=mu+LZ. Then Cov(X)=LIL-transpose=Sigma.

Common miss

The fastest way to get this chapter wrong is to use the visible procedure without naming its assumption, units, time period, reference group, or boundary. I put that condition beside the work and verify the result independently.

Solved chapter practice

Practice 1. How can Z from a standard multivariate normal produce covariance Sigma?
Solution. Find a Cholesky factor L with LL-transpose=Sigma and set X=mu+LZ. Then Cov(X)=LIL-transpose=Sigma.

Practice 2. Explain multivariate normal computation to a classmate using one definition, one concrete example, one assumption, and one failure mode.
Solution. Begin with the core claim above, use the worked case as the concrete example, state the condition highlighted by the warning, and choose a failure where that condition does not hold. A complete explanation connects all four pieces instead of listing them.

Mastery practice A. Reconstruct multivariate normal computation from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a multivariate normal computation problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent

check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 3 Principal components and singular value decomposition

The idea

PCA rotates centered data into orthogonal directions ordered by variance. SVD computes components stably and extends to rank-deficient or rectangular matrices; scaling and explained variance determine interpretation.

Rebuild the chapter from first principles

The claim I need to own is this: PCA rotates centered data into orthogonal directions ordered by variance. SVD computes components stably and extends to rank-deficient or rectangular matrices; scaling and explained variance determine interpretation. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of principal components and singular value decomposition. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes,

the invariant breaks, or the model stops matching reality.

- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach principal components and singular value decomposition on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from multivariate normal computation to clustering objectives and algorithms. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

The full working model

- **Core claim.** PCA rotates centered data into orthogonal directions ordered by variance.
- **What moves.** SVD computes components stably and extends to rank-deficient or rectangular matrices; scaling and explained variance determine interpretation.
- **What keeps the answer honest.** The model becomes useful only when I can apply principal components and singular value decomposition to an unfamiliar case and defend the assumptions.
- **Mastery standard.** I can define the objects, rebuild the rule, work a fresh case, compare alternatives, and diagnose a broken solution without relying on recognition.

How I actually think about it

PCA compresses variation in X, not necessarily information about an outcome I care about.

The easy version

PCA rotates centered data into orthogonal directions ordered by variance.

Worked example

Question. Why center data before PCA?
Walkthrough. Without centering, the first direction can chase the mean offset from the origin instead of variation around the mean. Standardization may also be needed when units differ.

Common miss

The fastest way to get this chapter wrong is to use the visible procedure without naming its assumption, units, time period, reference group, or boundary. I put that condition beside the work and verify the result independently.

Solved chapter practice

Practice 1. Why center data before PCA?
Solution. Without centering, the first direction can chase the mean offset from the origin instead of variation around the mean. Standardization may also be needed when units differ.

Practice 2. Explain principal components and singular value decomposition to a classmate using one definition, one concrete example, one assumption, and one failure mode.
Solution. Begin with the core claim above, use the worked case as the concrete example, state the condition highlighted by the warning, and choose a failure where that condition does not hold. A complete explanation connects all four pieces instead of listing them.

Mastery practice A. Reconstruct principal components and singular value decomposition from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a principal components and singular value decomposition problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 4 Clustering objectives and algorithms

The idea

K-means alternates assignment and centroid updates to reduce within-cluster squared distance. Hierarchical and model-based methods encode other structures. Initialization, local optima, distance, scaling, and stability checks govern conclusions.

Rebuild the chapter from first principles

The claim I need to own is this: K-means alternates assignment and centroid updates to reduce within-cluster squared distance. Hierarchical and model-based methods encode other structures. Initialization, local optima, distance, scaling, and stability checks govern conclusions. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of clustering objectives and algorithms. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach clustering objectives and algorithms on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from principal components and singular value decomposition to gradient, newton, and constrained optimization. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

The full working model

- **Core claim.** K-means alternates assignment and centroid updates to reduce within-cluster squared distance.
- **What moves.** Hierarchical and model-based methods encode other structures.

- **What keeps the answer honest.** Initialization, local optima, distance, scaling, and stability checks govern conclusions.
- **Mastery standard.** I can define the objects, rebuild the rule, work a fresh case, compare alternatives, and diagnose a broken solution without relying on recognition.

How I actually think about it

I run clustering many ways and look for stable, useful structure instead of treating one color map as truth.

The easy version

K-means alternates assignment and centroid updates to reduce within-cluster squared distance.

Worked example

Question. Why use k-means++ or multiple random starts?
Walkthrough. The objective is nonconvex and Lloyd's algorithm can settle in poor local optima. Better seeding and repeated starts improve the solution found.

Common miss

The fastest way to get this chapter wrong is to use the visible procedure without naming its assumption, units, time period, reference group, or boundary. I put that condition beside the work and verify the result independently.

Solved chapter practice

Practice 1. Why use k-means++ or multiple random starts?
Solution. The objective is nonconvex and Lloyd's algorithm can settle in poor local optima. Better seeding and repeated starts improve the solution found.

Practice 2. Explain clustering objectives and algorithms to a classmate using one definition, one concrete example, one assumption, and one failure mode.
Solution. Begin with the core claim above, use the worked case as the concrete example, state the condition highlighted by the warning, and choose a failure where that condition does not hold. A complete explanation connects all four pieces instead of listing them.

Mastery practice A. Reconstruct clustering objectives and algorithms from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a clustering objectives and algorithms problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent

check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 5 Gradient, Newton, and constrained optimization

The idea

Optimization algorithms use objectives, gradients, Hessians, line searches, trust regions, or constraints. Convexity separates local from global reasoning; scaling and stopping criteria determine practical convergence.

Rebuild the chapter from first principles

The claim I need to own is this: Optimization algorithms use objectives, gradients, Hessians, line searches, trust regions, or constraints. Convexity separates local from global reasoning; scaling and stopping criteria determine practical convergence. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of gradient, newton, and constrained optimization. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.

- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach gradient, newton, and constrained optimization on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from clustering objectives and algorithms to stochastic optimization and statistical loss. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

The full working model

- **Core claim.** Optimization algorithms use objectives, gradients, Hessians, line searches, trust regions, or constraints.
- **What moves.** Convexity separates local from global reasoning; scaling and stopping criteria determine practical convergence.
- **What keeps the answer honest.** The model becomes useful only when I can apply gradient, newton, and constrained optimization to an unfamiliar case and defend the assumptions.
- **Mastery standard.** I can define the objects, rebuild the rule, work a fresh case, compare alternatives, and diagnose a broken solution without relying on recognition.

How I actually think about it

I verify a gradient numerically on a toy input before blaming the optimizer.

The easy version

Optimization algorithms use objectives, gradients, Hessians, line searches, trust regions, or constraints.

Worked example

Question. What advantage does Newton's method gain from the Hessian?
Walkthrough. It uses local curvature to rescale the gradient, often converging rapidly near a well-behaved optimum, but Hessian cost and indefiniteness can cause problems.

Common miss

The fastest way to get this chapter wrong is to use the visible procedure without naming its assumption, units, time period, reference group, or boundary. I put that condition beside the

work and verify the result independently.

Solved chapter practice

Practice 1. What advantage does Newton's method gain from the Hessian?
Solution. It uses local curvature to rescale the gradient, often converging rapidly near a well-behaved optimum, but Hessian cost and indefiniteness can cause problems.

Practice 2. Explain gradient, newton, and constrained optimization to a classmate using one definition, one concrete example, one assumption, and one failure mode.
Solution. Begin with the core claim above, use the worked case as the concrete example, state the condition highlighted by the warning, and choose a failure where that condition does not hold. A complete explanation connects all four pieces instead of listing them.

Mastery practice A. Reconstruct gradient, newton, and constrained optimization from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a gradient, newton, and constrained optimization problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 6 Stochastic optimization and statistical loss

The idea

Large-data fitting uses stochastic and mini-batch gradients, momentum, adaptive steps, and regularization. Learning rates trade speed against stability; noisy gradients require convergence diagnostics and held-out assessment.

Rebuild the chapter from first principles

The claim I need to own is this: Large-data fitting uses stochastic and mini-batch gradients, momentum, adaptive steps, and regularization. Learning rates trade speed against stability; noisy gradients require convergence diagnostics and held-out assessment. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of stochastic optimization and statistical loss. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach stochastic optimization and statistical loss on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from gradient, newton, and constrained optimization to expectation-maximization and latent variables. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

The full working model

- **Core claim.** Large-data fitting uses stochastic and mini-batch gradients, momentum, adaptive steps, and regularization.
- **What moves.** Learning rates trade speed against stability; noisy gradients require convergence diagnostics and held-out assessment.

- **What keeps the answer honest.** The model becomes useful only when I can apply stochastic optimization and statistical loss to an unfamiliar case and defend the assumptions.
- **Mastery standard.** I can define the objects, rebuild the rule, work a fresh case, compare alternatives, and diagnose a broken solution without relying on recognition.

How I actually think about it

A jagged loss curve is not automatically failure; I ask whether the validation objective stabilizes at the required accuracy.

The easy version

Large-data fitting uses stochastic and mini-batch gradients, momentum, adaptive steps, and regularization.

Worked example

Question. What happens if a gradient-descent learning rate is too large?
Walkthrough. Updates can overshoot, oscillate, or diverge. Reduce or schedule the step size and inspect feature scaling and gradient magnitude.

Common miss

The fastest way to get this chapter wrong is to use the visible procedure without naming its assumption, units, time period, reference group, or boundary. I put that condition beside the work and verify the result independently.

Solved chapter practice

Practice 1. What happens if a gradient-descent learning rate is too large?
Solution. Updates can overshoot, oscillate, or diverge. Reduce or schedule the step size and inspect feature scaling and gradient magnitude.

Practice 2. Explain stochastic optimization and statistical loss to a classmate using one definition, one concrete example, one assumption, and one failure mode.
Solution. Begin with the core claim above, use the worked case as the concrete example, state the condition highlighted by the warning, and choose a failure where that condition does not hold. A complete explanation connects all four pieces instead of listing them.

Mastery practice A. Reconstruct stochastic optimization and statistical loss from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a stochastic optimization and statistical loss problem but never states a model or checks the result. Write the shortest useful review of that solution.

Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 7 Expectation-maximization and latent variables

The idea

EM alternates expected complete-data sufficient statistics with parameter maximization, increasing observed-data likelihood under regularity. Mixtures, missing data, initialization, local maxima, and label switching complicate interpretation.

Rebuild the chapter from first principles

The claim I need to own is this: EM alternates expected complete-data sufficient statistics with parameter maximization, increasing observed-data likelihood under regularity. Mixtures, missing data, initialization, local maxima, and label switching complicate interpretation. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of expectation-maximization and latent variables. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.

- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach expectation-maximization and latent variables on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from stochastic optimization and statistical loss to dynamic programming and reproducible algorithm design. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

The full working model

- **Core claim.** EM alternates expected complete-data sufficient statistics with parameter maximization, increasing observed-data likelihood under regularity.
- **What moves.** Mixtures, missing data, initialization, local maxima, and label switching complicate interpretation.
- **What keeps the answer honest.** The model becomes useful only when I can apply expectation-maximization and latent variables to an unfamiliar case and defend the assumptions.
- **Mastery standard.** I can define the objects, rebuild the rule, work a fresh case, compare alternatives, and diagnose a broken solution without relying on recognition.

How I actually think about it

My shorthand for this chapter is: the E-step fills in expected hidden structure; the M-step refits as if those expected counts were usable evidence.

The easy version

EM alternates expected complete-data sufficient statistics with parameter maximization, increasing observed-data likelihood under regularity.

Worked example

Question. Does EM guarantee the global maximum?
Walkthrough. No. It typically improves or preserves likelihood each iteration but can converge to a local maximum or saddle point, so multiple starts and diagnostics are important.

Common miss

The fastest way to get this chapter wrong is to use the visible procedure without naming its assumption, units, time period, reference group, or boundary. I put that condition beside the work and verify the result independently.

Solved chapter practice

Practice 1. Does EM guarantee the global maximum?
Solution. No. It typically improves or preserves likelihood each iteration but can converge to a local maximum or saddle point, so multiple starts and diagnostics are important.

Practice 2. Explain expectation-maximization and latent variables to a classmate using one definition, one concrete example, one assumption, and one failure mode.
Solution. Begin with the core claim above, use the worked case as the concrete example, state the condition highlighted by the warning, and choose a failure where that condition does not hold. A complete explanation connects all four pieces instead of listing them.

Mastery practice A. Reconstruct expectation-maximization and latent variables from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a expectation-maximization and latent variables problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 8 Dynamic programming and reproducible algorithm design

The idea

Dynamic programming solves overlapping subproblems by defining states, transitions, base cases, and an evaluation order. Statistical uses include sequence models and changepoints; complexity, memory, backtracking, and numerical scaling matter.

Rebuild the chapter from first principles

The claim I need to own is this: Dynamic programming solves overlapping subproblems by defining states, transitions, base cases, and an evaluation order. Statistical uses include sequence models and changepoints; complexity, memory, backtracking, and numerical scaling matter. I do not count that as

learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of dynamic programming and reproducible algorithm design. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach dynamic programming and reproducible algorithm design on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from expectation-maximization and latent variables to the cumulative course model. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

The full working model

- **Core claim.** Dynamic programming solves overlapping subproblems by defining states, transitions, base cases, and an evaluation order.
- **What moves.** Statistical uses include sequence models and changepoints; complexity, memory, backtracking, and numerical scaling matter.
- **What keeps the answer honest.** The model becomes useful only when I can apply dynamic programming and reproducible algorithm design to an unfamiliar case and defend the assumptions.
- **Mastery standard.** I can define the objects, rebuild the rule, work a fresh case, compare alternatives, and diagnose a broken solution without relying on recognition.

How I actually think about it

I write the recurrence and state meaning before coding loops; otherwise the table becomes unexplained bookkeeping.

The easy version

Dynamic programming solves overlapping subproblems by defining states, transitions, base cases, and an evaluation order.

Worked example

Question. What are the four parts of a dynamic-programming solution?
Walkthrough. Define the state, recurrence or transition, base cases, and computation order. Then store predecessor information if the optimal path itself must be reconstructed.

Common miss

The fastest way to get this chapter wrong is to use the visible procedure without naming its assumption, units, time period, reference group, or boundary. I put that condition beside the work and verify the result independently.

Solved chapter practice

Practice 1. What are the four parts of a dynamic-programming solution?
Solution. Define the state, recurrence or transition, base cases, and computation order. Then store predecessor information if the optimal path itself must be reconstructed.

Practice 2. Explain dynamic programming and reproducible algorithm design to a classmate using one definition, one concrete example, one assumption, and one failure mode.
Solution. Begin with the core claim above, use the worked case as the concrete example, state the condition highlighted by the warning, and choose a failure where that condition does not hold. A complete explanation connects all four pieces instead of listing them.

Mastery practice A. Reconstruct dynamic programming and reproducible algorithm design from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects

and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a dynamic programming and reproducible algorithm design problem but never states a model or checks the result. Write the shortest useful review of that solution.

Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

Cumulative study plan

I use this packet in three passes. First I reconstruct each model and work its examples without looking. Second I mix chapters so the tool is no longer named for me. Third I explain a complete case aloud, including assumptions, units, uncertainty, failure modes, and what evidence would change my conclusion.