CAI 4104/6108 – Machine Learning Engineering:

Gradient Descent

Prof. Vincent Bindschaedler

Spring 2024

Administrivia: Midterm

- Reminder: Midterm is coming up!
  - Monday 2/19 and Wednesday 2/21 during class time (10:40 - 11:30) in **FLG 0220**
    - Topics: everything until 2/16
  - Duration: 50 minutes
  - Schedule:
    - **CAI6108**: take the exam on **2/19**
    - **CAI4104**: take the exam on **2/21**
    - Lecture that week (topic: Unsupervised Learning) will be pre-recorded
  - Format: **closed-books**, (blank) scratch paper, and physical calculator are allowed (no phones!)
  - Questions: short answers + problems

- Reminder: Sample Midterm (Practice Questions):
  - On Canvas — 50 minutes (18 short answers) + 3 problems.
  - This is for you to practice. Do **not** overfit!

Reminder: Metrics for Classification

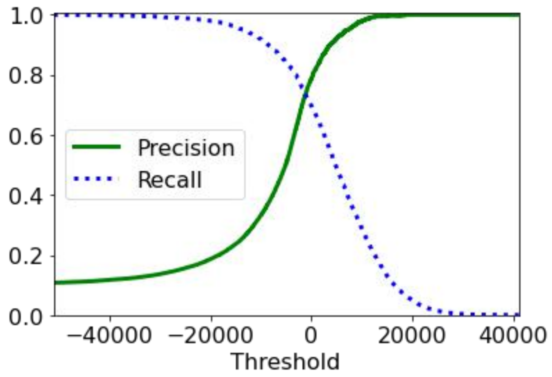
- Confusion Matrix
  - Applicable to classification in general
- Focus on the binary classification case:
  - One of the classes is designated as "*positive*" (the other is "*negative*")
  - False positives are Type I errors
    - Think of them as "false alarms"
  - False negatives are Type II errors
    - Think of them as "missed detections"
  - Prevalence (aka base rate):
    - Proportion of positive examples
- Baselines?
  - Statistical mode prediction
  - Random guessing

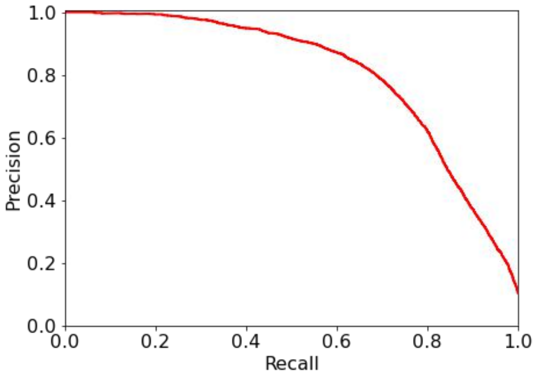
		Actual +	Actual -
Predicted	+	True Positive (TP)	False Positive (FP)
Predicted	-	False Negative (FN)	True Negative (TN)

- Accuracy: (TP + TN) / (TP + FP + TN + FN)
- Recall: TP / (TP + FN)
  - Also called: True positive rate (TPR), Sensitivity
- False negative rate (FNR): FN / (TP + FN) = 1 - TPR
- False positive rate (FPR): FP / (FP + TN)
- True negative rate (TNR): TN / (FP + TN) = 1 - FPR
  - Also called Specificity and Selectivity
- Precision: TP / (TP + FP)
  - Also called: Positive predictive value (PPV)
- False discovery rate (FDR): FP / (TP + FP) = 1 - Precision

Reminder: Precision-Recall Tradeoff

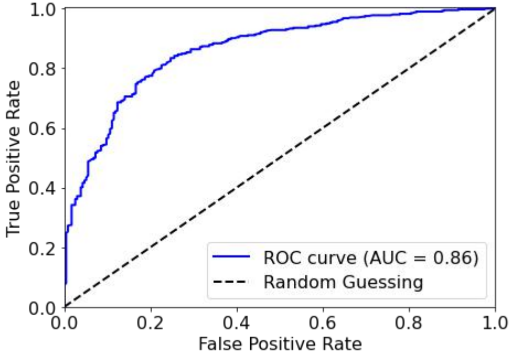
- Decision Threshold / Function
  - (In most cases) when you train a classification model, you actually train a family of classifiers!
    - The model assigns scores (or probabilities) to examples
    - Make predictions based on scores using a specific decision threshold
- The Tradeoff
  - The threshold provides a tradeoff between Type I and Type II errors
  - A popular way to express this tradeoff is Precision versus Recall
    - We need to choose between high precision and high recall
    - Q: What if we need to achieve precision above a specific value (e.g., 90%)?
- Optimizing and Statisficing
  - One way of navigating the tradeoff is to set a cutoff for precision or recall
  - E.g.: Pick the model with precision ≥95% that maximimizes recall



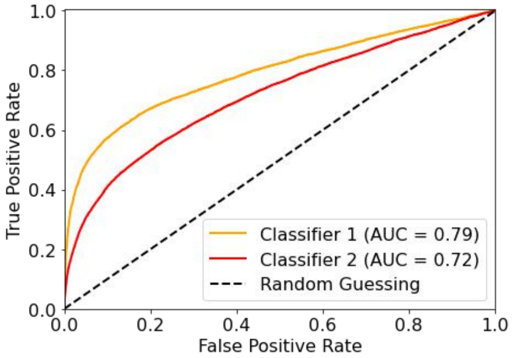


Reminder: Performance Curves

- Receiver Operating Characteristics (ROC)
  - Plots true positive rate (TPR) versus false positive rate (FPR)
    - How? Vary the threshold
  - Each point on the curve (a valid pair (TPR,FPR)) is a valid tradeoff point
    - E.g.: Equal Error Rate (EER) — point where FPR and FNR are equal



- Area Under Curve (AUC)
  - This is exactly the area under the ROC curve
    - AUC = 0 means the worst possible classifier
    - AUC = 0.5 is a random classifier
    - AUC = 1.0 is a perfect classifier



- Note: there are other performance curves
  - For example: Detection Error Tradeoff (DET) curves
    - False positive rate (FPR) versus the false negative rate (FNR), usually a log-log plot

Base Rate Fallacy

- Example:
  - Suppose you have a very accurate classifier (e.g., 99% accurate) to predict whether a person suffers from a specific disease D from features of their blood
  - Q: Should we test everyone in the world? Why or why not?
    - It depends on the base rate!

- Base rate fallacy / base rate neglect
  - Error in reasoning: confusing a classifier's prior probability of correct prediction and the posterior probability of a true positive
  - Suppose we have a classifier that has a false positive rate of 2% (i.e., 2% of the time it predicts '+' when the true label is '-') and a true positive rate of 100% (i.e., it never fails to detect '+' instances)
  - What is the probability that if the classifier predicts '+' the true label is in fact '+'?
    - If the base rate is 0.5 (i.e., 50% of instances are '+') then it is: ~98%
    - If the base rate is 0.001 (i.e., 0.1% of instances are '+') then it is: **~4.8%**

Base Rate Fallacy

- Base rate fallacy / base rate neglect
  - Suppose we have a classifier that has a false positive rate of 2% (i.e., 2% of the time it predicts ‘+’ when the true label is ‘-’) and a true positive rate of 100% (i.e., it never fails to detect ‘+’ instances)
  - What is the probability that if the classifier predicts ‘+’ the true label is in fact ‘+’?
    - If the base rate is 0.5 (i.e., 50% of instances are ‘+’) then it is: ~98%
    - If the base rate is 0.001 (i.e., 0.1% of instances are ‘+’) then it is: ~4.8%

		Actual	
		+	-
Predicted	+	5000	100
	-	0	4900

Pr(Actual + | Predicted +) = 5000/5100 ≈ 0.98

		Actual	
		+	-
Predicted	+	10	200
	-	0	9790

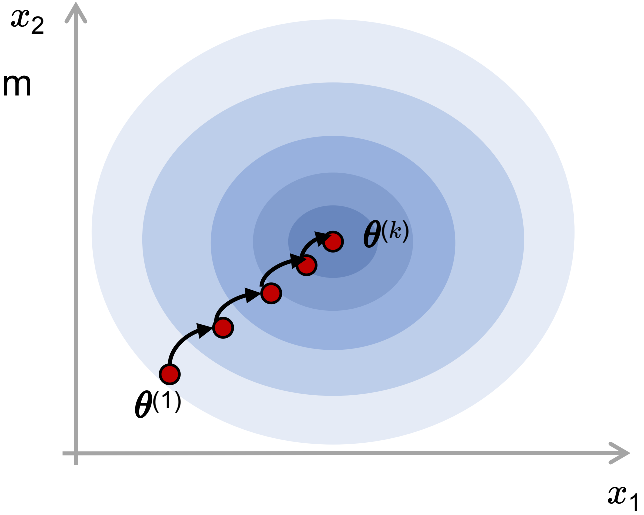
Pr(Actual + | Predicted +) = 10/210 ≈ 0.0476

- Note: there are other (similar) errors in reasoning
  - Examples: Prosecutor’s fallacy, Simpson’s paradox, etc.

Gradient Descent

- To solve an optimization problem, we can use Gradient Descent (or one of its variants)
  - Generic iterative procedure to update the parameters based on the **gradient**

- We want to minimize the loss function: $L(\boldsymbol{\theta})$
  - Here $\boldsymbol{\theta}$ are the parameters (weights)
  - η is the learning rate — size of steps we take towards the minimum
  - $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$ is the gradient of the loss with respect to the parameters

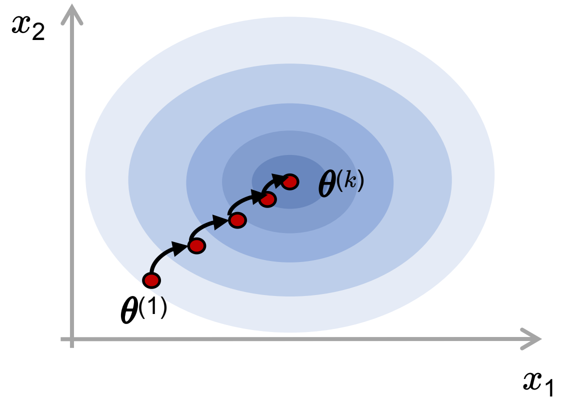


Gradient Descent

- Inputs: dataset $\boldsymbol{X}, \boldsymbol{y}$; model f, loss function L ; learning rate η >0 ; max number of iterations k
- Output: parameter vector $\boldsymbol{\theta}$
- Procedure:
  - Initialize $\boldsymbol{\theta}^{(1)}$ randomly
  - For j = 1, 2, ..., k-1
    - Take a batch of data $\boldsymbol{X}_B$, $\boldsymbol{y}_B$
    - Compute the gradient: $\boxed{\mathbf{g}=\nabla_{\theta} L(f(\boldsymbol{X}_B; \boldsymbol{\theta}^{(j)}), \boldsymbol{y}_B)}$
    - Update parameters: $\boxed{\boldsymbol{\theta}^{(j+1)} = \boldsymbol{\theta}^{(j)} - \eta\, \mathbf{g}}$
  - Output $\boldsymbol{\theta}^{(k)}$

This is the gradient of the loss L given current model parameters taken **with respect to** the parameters $\boldsymbol{\theta}$

Update parameters by taking steps of size η in the **opposite direction** of the gradient

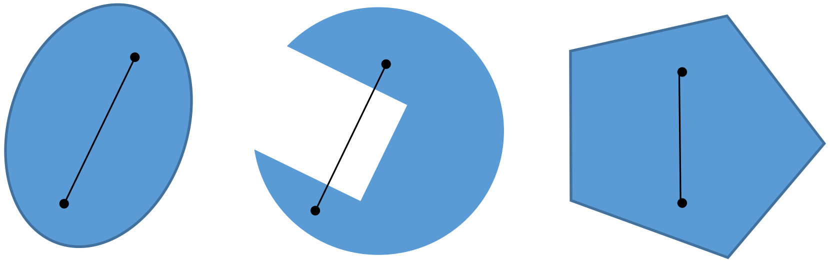


- Q: Are we guaranteed to converge to optimal parameters $\boldsymbol{\theta}^*$?

Reminder: Convex Sets & Convexity

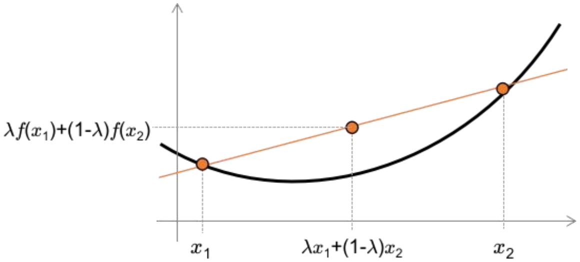
- Convex Sets:

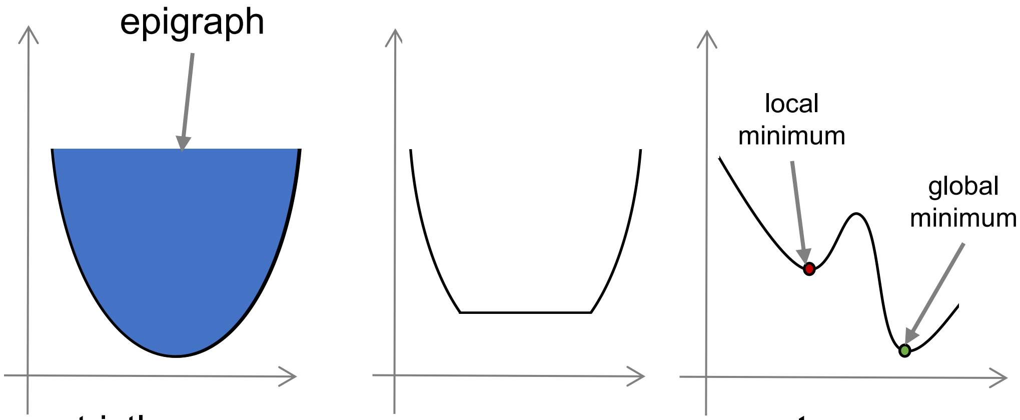
Set $\mathcal{X}$ is convex if for any $a, b \in \mathcal{X}$
the line λa+(1-λ)$b \in \mathcal{X}$ for $\lambda \in [0,1]$



- Convexity:

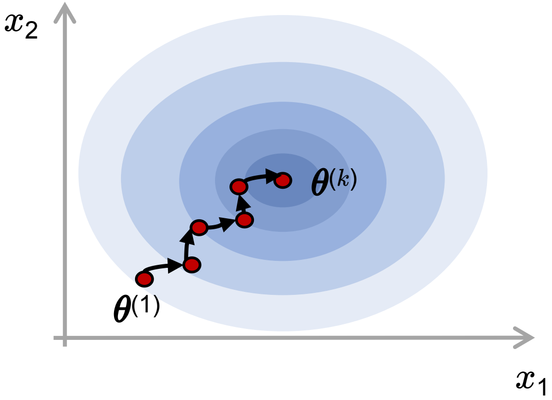
Function f on a convex set $\mathcal{X}$ is convex
if for any $x, y \in \mathcal{X}$ and $\lambda \in [0,1]$:
$\lambda f(x)+(1-\lambda) f(y) \geq f(\lambda x+(1-\lambda) y)$





Types of Gradient Descent

- Procedure:
  - Initialize $\boldsymbol{\theta}^{(1)}$ randomly
  - For j = 1, 2, ..., k-1
    - Take a batch of data $\boldsymbol{X}_B$, $\boldsymbol{y}_B$
    - Compute the gradient $\mathbf{g}=\nabla_{\boldsymbol{\theta}} L(f(\boldsymbol{X}_B; \boldsymbol{\theta}^{(j)}), \boldsymbol{y}_B)$
    - Update parameters: $\boldsymbol{\theta}^{(j+1)} = \boldsymbol{\theta}^{(j)} - \eta\, \mathbf{g}$
  - Output $\boldsymbol{\theta}^{(k)}$



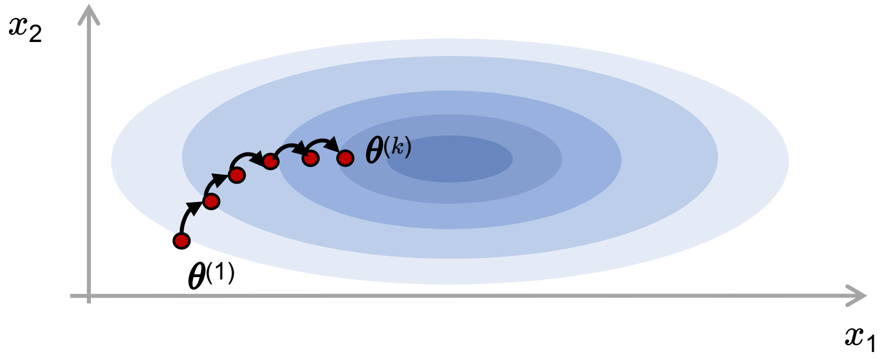
- Batch Gradient Descent (GD)
  - In each iteration: calculate the gradient of the entire dataset [i.e., $(\boldsymbol{X}_B, \boldsymbol{y}_B) = (\boldsymbol{X}, \boldsymbol{y})$]
- Stochastic ("on-line") Gradient Descent
  - In each iteration, randomly pick a single example and calculate its gradient [i.e., $(\boldsymbol{X}_B, y_B) = (\boldsymbol{x}_i, y_i)$]
- Mini-batch Stochastic Gradient Descent ("SGD")
  - Randomly shuffle the entire dataset and process it in mini-batches steps [i.e., $(\boldsymbol{X}_B, y_B)$ is a mini-batch of m examples]
    - One **epoch** is the number of iterations to process the entire dataset
  - What we typically use in practice: has stochasticity but does **not** require computing the gradient over the entire dataset

Important Considerations

- Major consideration: learning schedule
  - Way to set the **learning rate** in each iteration
    - E.g.: constant, time-based decay, step decay, exponential decay, etc.

- Variants / Optimizers:
  - AdaGrad, RMSProp, Adam, etc.

- Feature scaling: we **should** rescale features before doing gradient descent
  - What happens if we don't?
    - Convergence may be slow...



Illustrated Example

- Let's train a linear regression model using gradient descent
  - Loss: Mean Squared Error (MSE)
  - MSE($\boldsymbol{w}$,b) := $1/n \sum_i [\boldsymbol{w}\, \boldsymbol{x_i} + b - y_i]^2 = 1/n \sum_i [\boldsymbol{\theta}\, \boldsymbol{x_i} - y_i]^2$ = MSE($\boldsymbol{\theta}$)
    - Here: $\boldsymbol{\theta} = (b, \boldsymbol{w})$ and we extend $\boldsymbol{X}$ to incorporate a constant feature ($\boldsymbol{x_{i,0}}$=1 for all i)

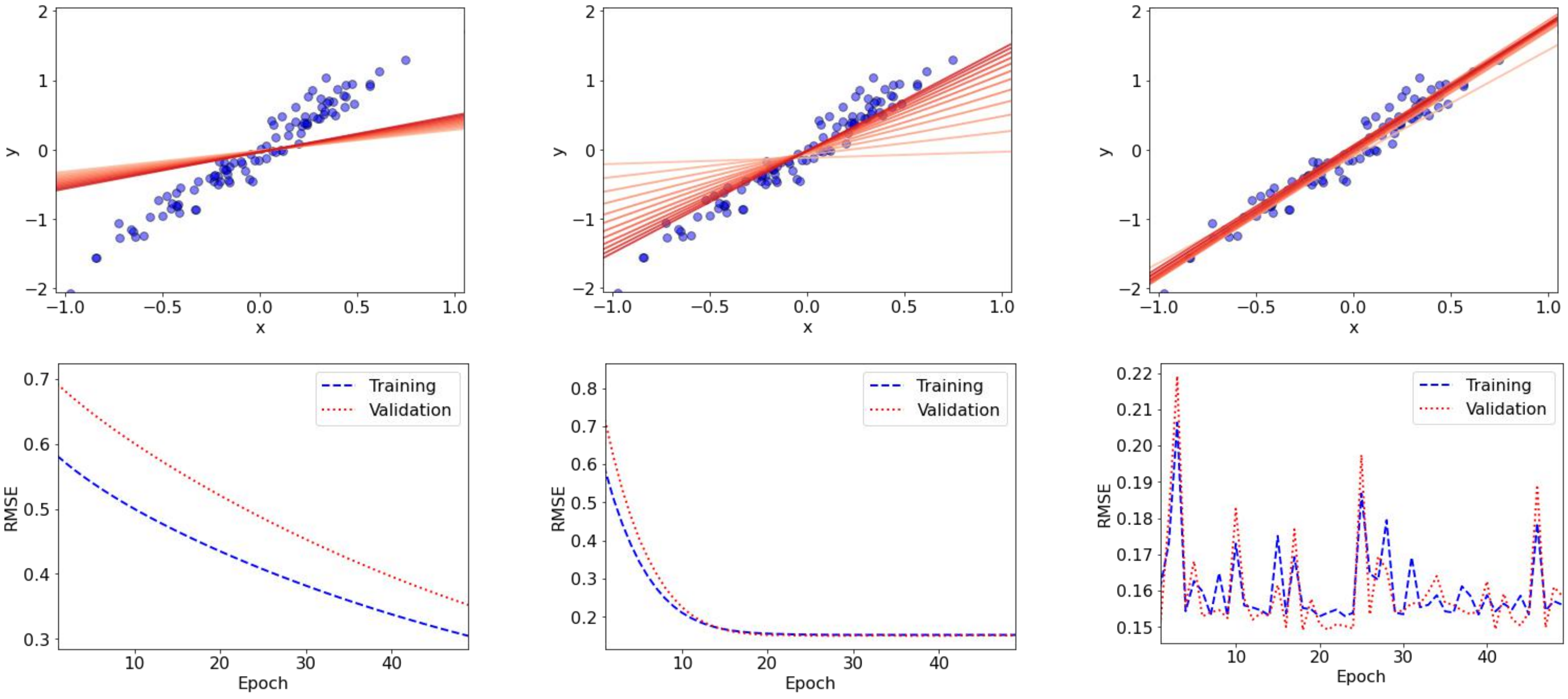
- Gradient step: $\boldsymbol{\theta}^{(j+1)} = \boldsymbol{\theta}^{(j)} - \eta \nabla_{\boldsymbol{\theta}}$ MSE($\boldsymbol{\theta}$)
  - η is the learning rate (hyperparameter)

- What is the gradient of MSE($\boldsymbol{\theta}$)?
  - $\nabla_{\boldsymbol{\theta}}$ MSE($\boldsymbol{\theta}$) = $2/n\, \boldsymbol{X}^T (\boldsymbol{\theta}\, \boldsymbol{X} - \boldsymbol{y})$
  - Q: What about the gradient for ridge regression?

Learning Rate?

- How do we set the learning rate?

η = 0.005 (too small) η = 0.05 (just right) η = 0.5 (too large)



Next Time

- Friday (2/16): Exercise and (or) Q&A

- Upcoming:
  - Midterm on 2/19 and 2/21