

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	6	화면설명
<div> <div>➤Intro</div> <div> <div>•학습열기</div> <div>•학습목표</div> </div> <div>➤ 학습하기</div> <div> <div>1. DNN 학습 최적화</div> <div>2. DNN 구조 최적화</div> <div>3. Dataset 최적화</div> </div> <div>➤ 적용하기</div> <div>➤ Outro</div> <div>•문제풀기</div> </div>	<div> <div>• 모델 성능을 끌어올리는 비밀 무기</div> <p>여러분은 혹시 이런 경험 있으신가요? 같은 데이터를 사용했는데 어떤 모델은 높은 정확도를 보이고, 어떤 모델은 전혀 기대 이하의 결과를 내는 경우 말입니다. 이 차이는 바로 '학습 방법'의 차이에서 비롯됩니다. 이번 시간에는 인공지능 모델의 성능을 획기적으로 끌어올릴 수 있는 4가지 기술에 대해 살펴보려 합니다.첫째는 DNN(Deep Neural Network), 즉 깊은 신경망 구조입니다. 이 구조는 복잡한 패턴을 정교하게 학습할 수 있게 해주지만, 때로는 오히려 학습을 방해하기도 합니다. 둘째는 배치 정규화(Batch Normalization)로, 학습의 안정성과 속도를 높여주는 중요한 기법입니다.셋째는 데이터의 균형을 맞추는 샘플링(Sampling) 기술입니다. 클래스가 불균형한 데이터셋에서는 소수 클래스에 대한 학습이 어려워 성능이 떨어지는데, 이를 해결하는 대표적인 방법이죠. 마지막으로는 데이터 증강(Data Augmentation)입니다. 부족한 데이터를 '창의적으로' 확장해 모델이 다양한 상황에 잘 적응할 수 있도록 돕는 전략입니다.이러한 기법을 적절히 조합하면, 단순한 모델조차도 놀라운 성능 향상을 이끌어낼 수 있습니다. 이번 학습을 통해, 그 '비밀 무기'들을 함께 배워보도록 하겠습니다.</p> </div>				<div> <div>① 본 학습 내용으로 들어가기 전, 학습 주제의 흥미를 이끌 만한 도입부의 내용이 있다면 제시해주세요.</div> <div>② ex. 관련 뉴스기사, 실생활과 관련된 이야기 등</div> <div>③ 저작권 침해가 되지 않도록 내용을 구성해 주세요.</div> <div>④ 출처가 있을 경우 반드시 작성해 주세요.</div> </div>
					용어설명
내레이션					3

과정명		PyTorch로 배우는 머신러닝 알고리즘		회차명	6	화면설명	
➤Intro •학습열기 •학습목표  ➤학습하기 1. DNN 학습 최적화 2. DNN 구조 최적화 3. Dataset 최적화  ➤적용하기  ➤Outro •문제풀기	<div>◆ 학습목표</div> <div>1. 학습률 조정 및 정규화 기법을 적용할 수 있다.</div> <div>2. 은닉층 개수와 뉴런 수를 조정하여 성능을 비교할 수 있다.</div> <div>3. 데이터 증강 및 샘플링 기법을 적용할 수 있다.</div> <div>◆ 학습내용</div> <div>1. DNN 학습 최적화</div> <div>2. DNN 구조 최적화</div> <div>3. Dataset 최적화</div>					① 학습내용과 학습목표는 강의계획서와 일치해야 하며, 필요시 강의계획서를 수정할 수 있습니다.	
						② 학습목표	✓ 각 레슨에 맞는 학습목표를 2~3개 작성해 주세요.
						③ 학습내용	✓ 1회차 당 25분 분량이 되도록 2~3개 레슨으로 구성해주세요.  ✓ 학습내용과 레슨명은 일치해야 합니다.
<div>용어설명</div>							
내레이션	4						

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	6	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤학습하기</div> <div>1. DNN 학습 최적화</div> <div>2. DNN 구조 최적화</div> <div>3. Dataset 최적화</div> <div>➤적용하기</div> <div>➤Outro</div> <div>•문제풀기</div>	<div>간지</div> <div>DNN 학습 최적화</div>			
				용어설명
내레이션	5			

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	6	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤학습하기</div> <div>1. DNN 학습 최적화</div> <div>2. DNN 구조 최적화</div> <div>3. Dataset 최적화</div> <div>➤적용하기</div> <div>➤Outro</div> <div>•문제풀기</div>	<div>• 학습률과 스케줄러 개요</div> <div>학습률이란?</div> <div>학습률은 가중치 파라미터를 얼마나 빠르게 바꿀지를 결정하는 하이퍼파라미터</div> <div>너무 크면 발산, 너무 작으면 수렴이 느림 → 속도와 안정성 모두 영향을 줌</div> <div>고정 학습률의 문제점</div> <div>초기에 크면 빠르게 내려가지만 나중에 안정적이지 못하고 요동침</div> <div>작으면 안정적이나 학습 시간이 오래 걸림</div> <div>그래서 스케줄러가 필요</div> <div>스케줄러의 역할: 학습률을 "시간에 따라 똑똑하게" 조정</div> <div>초기엔 빠르게 움직이고, 후반에는 세밀하게 조정하려는 전략</div> <div>학습 후반에는 너무 큰 학습률이면 최솟값을 지나쳐버릴 위험</div> <div>과적합을 줄이고, 수렴을 빠르게 만들</div>			용어설명

내레이션

6

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	6	화면설명								
과정명	➤Intro	<div>• 학습률 초기값</div> <div>lr(learning rate): 파라미터 업데이트의 "속도"</div> <div>너무 크면: 손실 함수가 요동치거나 발산함</div> <div>너무 작으면: 수렴 속도가 느려지고, 지역 최솟값에서 탈출 못 함</div> <table><tr><th>모델 규모</th><th>추천 lr</th></tr><tr><td>소규모 MLP</td><td>0.001 ~ 0.01</td></tr><tr><td>복잡한 MLP</td><td>0.0005 ~ 0.005</td></tr><tr><td>정규화 강하게 했을 때</td><td>0.01 이상도 가능</td></tr></table>			모델 규모	추천 lr	소규모 MLP	0.001 ~ 0.01	복잡한 MLP	0.0005 ~ 0.005	정규화 강하게 했을 때	0.01 이상도 가능	
	모델 규모				추천 lr								
	소규모 MLP				0.001 ~ 0.01								
	복잡한 MLP				0.0005 ~ 0.005								
정규화 강하게 했을 때	0.01 이상도 가능												
•학습열기													
•학습목표													
➤학습하기													
1. DNN 학습 최적화													
2. DNN 구조 최적화													
3. Dataset 최적화													
➤적용하기		용어설명											
➤Outro													
•문제풀기													
내레이션					7								

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	6	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤학습하기</div> <div>1. DNN 학습 최적화</div> <div>2. DNN 구조 최적화</div> <div>3. Dataset 최적화</div> <div>➤적용하기</div> <div>➤Outro</div> <div>•문제풀기</div>	<div>• 옵티마이저: optim.AdamW() Adam 옵티마이저의 개선 버전으로, 일반화 성능 향상과 과적합 완화에 효과적</div> <div>• 스케줄러 StepLR: 학습률을 단계적으로 감소 수렴을 빠르게 유도하고 지역 정답(local minima)을 더 효과적으로 찾을 수 있게 도와 줌 10 epoch마다 학습률을 0.5배로 감소시키는 학습률 스케줄러 step_size마다 학습률을 gamma배로 줄임 예: 0.01 → 0.005 → 0.0025 ... (10, 20, 30 epoch마다) 간단한 모델이나 데이터에 적합: 심장병 판별</div> <div># 7. 손실함수와 옵티마이저 criterion = nn.BCEWithLogitsLoss() # 옵티마이저 변경 optimizer = optim.AdamW(model.parameters(), lr=0.001) # 스케줄러 변경 scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=10, gamma=0.8)</div>			
				용어설명

내레이션

8

과정명		PyTorch로 배우는 머신러닝 알고리즘		회차명	6	화면설명																								
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤ 학습하기</div> <div>1. DNN 학습 최적화</div> <div>2. DNN 구조 최적화</div> <div>3. Dataset 최적화</div> <div>➤ 적용하기</div> <div>➤ Outro</div> <div>•문제풀기</div>	<div>• 스케줄러 비교</div> <table><tr><th>Scheduler</th><th>조정 방식</th><th>특징</th><th>추천 대상</th></tr><tr><td>StepLR</td><td>일정 에폭마다 감소</td><td>단순, 직관적</td><td>초보자용, 기본</td></tr><tr><td>ReduceLROnPlateau</td><td>성능 정체 시 감소</td><td>val_loss 기준, 적응형</td><td>중급자 이상</td></tr><tr><td>CosineAnnealingLR</td><td>코사인 곡선으로 감소</td><td>부드러운 감쇠</td><td>중~장기 학습</td></tr><tr><td>ExponentialLR</td><td>지수적으로 계속 감소</td><td>빠르게 줄이고 싶을 때</td><td>정밀한 튜닝 시</td></tr><tr><td>OneCycleLR</td><td>오르고 급히 떨어짐</td><td>빠른 수렴, 과감한 최적화 전략</td><td>고급자용, 대규모</td></tr></table>					Scheduler	조정 방식	특징	추천 대상	StepLR	일정 에폭마다 감소	단순, 직관적	초보자용, 기본	ReduceLROnPlateau	성능 정체 시 감소	val_loss 기준, 적응형	중급자 이상	CosineAnnealingLR	코사인 곡선으로 감소	부드러운 감쇠	중~장기 학습	ExponentialLR	지수적으로 계속 감소	빠르게 줄이고 싶을 때	정밀한 튜닝 시	OneCycleLR	오르고 급히 떨어짐	빠른 수렴, 과감한 최적화 전략	고급자용, 대규모	
	Scheduler	조정 방식	특징	추천 대상																										
	StepLR	일정 에폭마다 감소	단순, 직관적	초보자용, 기본																										
	ReduceLROnPlateau	성능 정체 시 감소	val_loss 기준, 적응형	중급자 이상																										
	CosineAnnealingLR	코사인 곡선으로 감소	부드러운 감쇠	중~장기 학습																										
	ExponentialLR	지수적으로 계속 감소	빠르게 줄이고 싶을 때	정밀한 튜닝 시																										
	OneCycleLR	오르고 급히 떨어짐	빠른 수렴, 과감한 최적화 전략	고급자용, 대규모																										

과정명		PyTorch로 배우는 머신러닝 알고리즘		회차명	6	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤학습하기</div> <div>1. DNN 학습 최적화</div> <div>2. DNN 구조 최적화</div> <div>3. Dataset 최적화</div> <div>➤적용하기</div> <div>➤Outro</div> <div>•문제풀기</div>	<div>간지</div> <div>DNN 구조 최적화</div>					
						용어설명
내레이션	10					



과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	6	화면설명
➤Intro  •학습열기  •학습목표  ➤학습하기 1. DNN 학습 최적화 2. DNN 구조 최적화 3. Dataset 최적화      ➤적용하기  ➤Outro  •문제풀기	<ul style="list-style-type: none"><li>DNN의 은닉층을 유연하게 구성하기 위한 클래스 DNN_Vriant() 생성</li></ul> <p>은닉층의 개수와 뉴런 수를 실험적으로 바꿔가며 모델 구조를 테스트할 수 있도록 설계</p> <pre># 7. DNN 모델 정의를 유연하게 정의 class DNN_Variant(nn.Module):     def __init__(self, input_dim, hidden_dims):         super().__init__()         layers = []         # input_dim → hidden_dim 연결         for dim in hidden_dims:             layers += [                 nn.Linear(input_dim, dim), # 선형 변환 (Fully Connected)                 nn.BatchNorm1d(dim),       # 배치 정규화 → 학습 안정화                 nn.ReLU(),                  # 비선형 활성화 → 복잡한 함수 근사                 nn.Dropout(0.3)             # 30% 확률로 뉴런 끄기 → 과적합 방지             ]             # 이후 다음 레이어에서는 input_dim = dim으로 갱신해서 이어짐             input_dim = dim         # 마지막 출력은 이진 분류이므로 노드 1개 (시그모이드는 이후에 적용됨)         layers += [nn.Linear(input_dim, 1)]         # 앞서 만든 레이어 리스트를 하나의 순차적 블록으로 구성         self.net = nn.Sequential(*layers)      def forward(self, x):         return self.net(x)</pre>				
					용어설명
내레이션	11				

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	6	화면설명
과정명	➤Intro	<ul style="list-style-type: none"><li>구조 변경 모델 생성</li></ul> <p>원하는 은닉층 구성 가능</p> <p>심장병 데이터는 단순한 모델과 적은 데이터이므로 2개의 은닉층 정도로 구성</p> <pre># 구조 변경 모델 생성 # 은닉층 1개, 뉴런 수 64 실험 # model = DNN_Variant(input_dim=X.shape[1], hidden_dims=[64]) # 1층 # 은닉층 2개, 뉴런 수 128, 64 실험 model = DNN_Variant(input_dim=X.shape[1], hidden_dims=[128, 64]) # 2층 # 은닉층 3개, 뉴런 수 256, 128, 64 실험 # model = DNN_Variant(input_dim=X.shape[1], hidden_dims=[256, 128, 64]) # 3층</pre>			
	•학습열기				
	•학습목표				
과정명	➤학습하기				
	1. DNN 학습 최적화				
	2. DNN 구조 최적화				
과정명	3. Dataset 최적화				
과정명	➤적용하기				용어설명
	➤Outro				
	•문제풀기				
내레이션		12			

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	6	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤ 학습하기</div> <div>1. DNN 학습 최적화</div> <div>2. DNN 구조 최적화</div> <div>3. Dataset 최적화</div> <div>➤ 적용하기</div> <div>➤ Outro</div> <div>•문제풀기</div>	<div>• 배치 정규화(batch normalization) 레이어</div> <div>딥러닝 학습의 안정성, 속도, 일반화 성능을 높이는 데 매우 중요한 핵심 기술</div> <div>초기 가중치에 덜 민감하고, 더 큰 학습률을 사용할 수 있음</div> <div>각 배치마다 입력 데이터를 정규화해서 평균이 0, 분산이 1에 가깝도록 만든 후 다시 학습 가능한 (원본 데이터와 유사하게) scale과 shift를 적용하는 기법</div> <div>scale → 정규화된 값을 “늘려주는” 과정</div> <div>shift → 정규화된 값을 “이동시켜주는” 과정</div>			
				용어설명
내레이션	13			

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	6	화면설명										
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤ 학습하기</div> <div>1. DNN 학습 최적화</div> <div>2. DNN 구조 최적화</div> <div>3. Dataset 최적화</div> <div>➤적용하기</div> <div>➤Outro</div> <div>•문제풀기</div>	<div>• BatchNorm을 사용하는 이유</div> <table><tr><th>사용 이유</th><th>설명</th></tr><tr><td>1. 학습 안정화</td><td>각 층의 입력 분포를 일정하게 유지하여 그라디언트(Gradient) 폭주/소실을 완화함</td></tr><tr><td>2. 빠른 수렴</td><td>학습률을 더 크게 설정해도 안정적으로 빠르게 수렴할 수 있음</td></tr><tr><td>3. 과적합 억제</td><td>배치 정규화가 노이즈 효과를 주어 Dropout과 유사한 regularization 역할 수행</td></tr><tr><td>4. 초기값 민감도 감소</td><td>가중치 초기화에 덜 민감해지고, 학습 시작이 더 유연해짐</td></tr></table> <pre># input_dim → hidden_dim 연결 for dim in hidden_dims:     layers += [         nn.Linear(input_dim, dim), # 선형 변환 (Fully Connected)         nn.BatchNorm1d(dim),       # 배치 정규화 → 학습 안정화         nn.ReLU(),                 # 비선형 활성화 → 복잡한 함수 근사         nn.Dropout(0.3)            # 30% 확률로 뉴런 끄기 → 과적합 방지     ] # 이후 다음 레이어에서는 input_dim = dim으로 갱신해서 이어짐 input_dim = dim</pre>			사용 이유	설명	1. 학습 안정화	각 층의 입력 분포를 일정하게 유지하여 그라디언트(Gradient) 폭주/소실을 완화함	2. 빠른 수렴	학습률을 더 크게 설정해도 안정적으로 빠르게 수렴할 수 있음	3. 과적합 억제	배치 정규화가 노이즈 효과를 주어 Dropout과 유사한 regularization 역할 수행	4. 초기값 민감도 감소	가중치 초기화에 덜 민감해지고, 학습 시작이 더 유연해짐	
	사용 이유	설명												
	1. 학습 안정화	각 층의 입력 분포를 일정하게 유지하여 그라디언트(Gradient) 폭주/소실을 완화함												
	2. 빠른 수렴	학습률을 더 크게 설정해도 안정적으로 빠르게 수렴할 수 있음												
3. 과적합 억제	배치 정규화가 노이즈 효과를 주어 Dropout과 유사한 regularization 역할 수행													
4. 초기값 민감도 감소	가중치 초기화에 덜 민감해지고, 학습 시작이 더 유연해짐													
			용어설명											

과정명		PyTorch로 배우는 머신러닝 알고리즘		회차명	6	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤ 학습하기</div> <div>1. DNN 학습 최적화</div> <div>2. DNN 구조 최적화</div> <div>3. Dataset 최적화</div>	<div><div>간지</div><div>Dataset 최적화</div></div>					
	<div>➤적용하기</div> <div>➤ Outro</div> <div>•문제풀기</div>					
내레이션	15					

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	6	화면설명								
➤Intro •학습열기 •학습목표  ➤학습하기 1. DNN 학습 최적화 2. DNN 구조 최적화 3. Dataset 최적화	<ul style="list-style-type: none"><li>• 데이터 증강 (Data Augmentation) 기존 데이터를 변형하거나 확장하여, 새로운 데이터를 생성하는 방법 데이터의 양을 늘리고 다양성도 확보함 주로 과적합 방지, 일반화 성능 향상 목적</li><li>• 샘플링 (Sampling) 특정 클래스를 기준으로 데이터를 인위적으로 줄이거나 늘리는 작업 타겟 클래스 간 데이터 비율이 불균형한 경우 사용됨 분류 문제에서 성능 왜곡 방지 목적</li></ul> <table><tr><th>기법 유형</th><th>설명</th></tr><tr><td>Oversampling</td><td>소수 클래스 샘플 수를 늘림 (ex: SMOTE)</td></tr><tr><td>Undersampling</td><td>다수 클래스 샘플 수를 줄임</td></tr><tr><td>Mixed Sampling</td><td>둘을 조합 (ex: SMOTE + TomekLinks)</td></tr></table>				기법 유형	설명	Oversampling	소수 클래스 샘플 수를 늘림 (ex: SMOTE)	Undersampling	다수 클래스 샘플 수를 줄임	Mixed Sampling	둘을 조합 (ex: SMOTE + TomekLinks)	
					기법 유형	설명							
					Oversampling	소수 클래스 샘플 수를 늘림 (ex: SMOTE)							
Undersampling	다수 클래스 샘플 수를 줄임												
Mixed Sampling	둘을 조합 (ex: SMOTE + TomekLinks)												
➤적용하기  ➤Outro •문제풀기													
					용어설명								
내레이션					16								

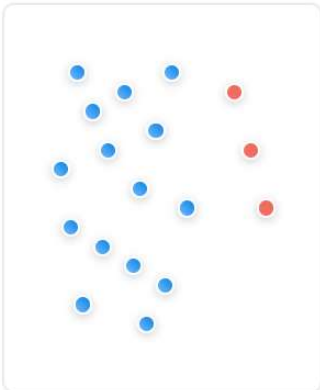
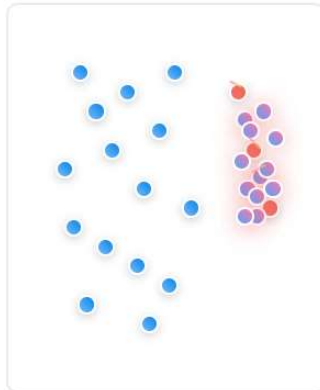
과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	6	화면설명																					
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤ 학습하기</div> <div>1. DNN 학습 최적화</div> <div>2. DNN 구조 최적화</div> <div>3. Dataset 최적화</div> <div>➤적용하기</div> <div>➤ Outro</div> <div>•문제풀기</div>		• 데이터 증강과 샘플링 비교 요약표																								
		<table><tr><th>항목</th><th>데이터 증강</th><th>샘플링</th></tr><tr><td>목적</td><td>다양성 확보, 일반화</td><td>클래스 불균형 해결</td></tr><tr><td>클래스 비율 변화</td><td>변화 없음</td><td>변화 있음</td></tr><tr><td>대표 기법</td><td>Noise 추가, 이미지 변형</td><td>SMOTE, RandomOverSampler</td></tr><tr><td>적용 시점</td><td>학습 데이터 전처리 or 실시간</td><td>학습 데이터 전처리 (fit_resample)</td></tr><tr><td>사용 대상</td><td>모든 클래스</td><td>주로 소수 클래스</td></tr><tr><td>리스크</td><td>과도한 왜곡은 오히려 과적합 유발</td><td>synthetic 데이터가 의미 없을 수 있음</td></tr></table>			항목	데이터 증강	샘플링	목적	다양성 확보, 일반화	클래스 불균형 해결	클래스 비율 변화	변화 없음	변화 있음	대표 기법	Noise 추가, 이미지 변형	SMOTE, RandomOverSampler	적용 시점	학습 데이터 전처리 or 실시간	학습 데이터 전처리 (fit_resample)	사용 대상	모든 클래스	주로 소수 클래스	리스크	과도한 왜곡은 오히려 과적합 유발	synthetic 데이터가 의미 없을 수 있음	
		항목	데이터 증강	샘플링																						
		목적	다양성 확보, 일반화	클래스 불균형 해결																						
		클래스 비율 변화	변화 없음	변화 있음																						
		대표 기법	Noise 추가, 이미지 변형	SMOTE, RandomOverSampler																						
		적용 시점	학습 데이터 전처리 or 실시간	학습 데이터 전처리 (fit_resample)																						
		사용 대상	모든 클래스	주로 소수 클래스																						
		리스크	과도한 왜곡은 오히려 과적합 유발	synthetic 데이터가 의미 없을 수 있음																						
					용어설명																					
내레이션	17																									

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	6	화면설명										
과정소개	➤Intro	• 오버샘플링을 위한 클래스 SMOTE 패키지 imbalanced-learn 필요, Colab에는 이미 설치됨 샘플링의 SMOTE(Synthetic Minority Oversampling Technique) 함수 소수 클래스 합성 과샘플링(오버샘플링) 기법 불균형 데이터셋에서 소수 클래스의 합성 샘플을 생성하여 클래스 균형을 맞추는 기법 가까운 이웃(k-NN)과 보간(interpolation) 방법을 사용해서 새로운 합성 샘플(synthetic sample)을 생성													
	•학습열기 •학습목표														
학습내용	➤학습하기														
	1. DNN 학습 최적화 2. DNN 구조 최적화 3. Dataset 최적화														
실용성	➤적용하기	<table><tr><th>단어</th><th>의미</th></tr><tr><td>Synthetic</td><td>인공적으로 생성된 (합성된)</td></tr><tr><td>Minority</td><td>소수 클래스 (불균형한 클래스 중 적은 쪽)</td></tr><tr><td>Oversampling</td><td>데이터 수를 인위적으로 늘림</td></tr><tr><td>Technique</td><td>기법, 방법론</td></tr></table>			단어	의미	Synthetic	인공적으로 생성된 (합성된)	Minority	소수 클래스 (불균형한 클래스 중 적은 쪽)	Oversampling	데이터 수를 인위적으로 늘림	Technique	기법, 방법론	용어설명
	단어				의미										
Synthetic	인공적으로 생성된 (합성된)														
Minority	소수 클래스 (불균형한 클래스 중 적은 쪽)														
Oversampling	데이터 수를 인위적으로 늘림														
Technique	기법, 방법론														
	➤Outro														
	•문제풀기														

내레이션

18



과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	6	화면설명
<div>➤Intro<ul style="list-style-type: none"><li>•학습열기</li><li>•학습목표</li></ul></div> <div>➤ 학습하기<ol style="list-style-type: none"><li>1. DNN 학습 최적화</li><li>2. DNN 구조 최적화</li><li>3. Dataset 최적화</li></ol></div> <div>➤ 적용하기</div> <div>➤ Outro<ul style="list-style-type: none"><li>•문제풀기</li></ul></div>	<div>• 오버샘플링의 이해</div> <div><div><div><div>불균형 데이터 → 균형 데이터 변환</div><div><div><div>원본 불균형 데이터셋</div><div></div><div>SMOTE 적용 후 데이터셋</div><div></div></div><div>→</div><div><div><div>● 다수 클래스 (Majority Class)</div><div>● 소수 클래스 (Minority Class)</div><div>● SMOTE 합성 샘플</div></div></div></div></div></div><div>강환수, 직접 그림</div></div>			
			용어설명	

내레이션

19

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	6	화면설명	
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤ 학습하기</div> <div>1. DNN 학습 최적화</div> <div>2. DNN 구조 최적화</div> <div>3. Dataset 최적화</div> <div>➤ 적용하기</div> <div>➤ Outro</div> <div>•문제풀기</div>	<div>• SMOTE 클래스 테스트</div> <div>일반적인 복제 방식과 달리, 새로운 데이터 포인트를 생성</div> <div>SMOTE는 소수 클래스에 대해서만 합성 샘플(synthetic sample)을 생성</div> <div>클래스는 절대 건드리지 않음</div> <div>클래스 비율을 1:1로 맞추는 것이 기본 설정</div> <div><pre>from imblearn.over_sampling import SMOTE from collections import Counter  X = [[i] for i in range(50)] y = [0]*40 + [1]*10  print("Before SMOTE:", Counter(y)) smote = SMOTE(random_state=42) X_res, y_res = smote.fit_resample(X, y) print("After SMOTE :", Counter(y_res))  Before SMOTE: Counter({0: 40, 1: 10}) After SMOTE : Counter({0: 40, 1: 40})</pre></div>				

내레이션

20

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	6	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤학습하기</div> <div>1.DNN 학습 최적화</div> <div>2.DNN 구조 최적화</div> <div>3.Dataset 최적화</div>	<div>• 심장병 데이터셋에서 오버샘플링</div> <div>심장병 유무 개수를 동일하게 조정</div> <div># 3. 훈련/테스트 분할</div> <div>from sklearn.model_selection import train_test_split</div> <div>X_train, X_test, y_train, y_test = train_test_split(X.values, y.values, test_size=0.2, random_state=42, stratify=y)</div> <div>)</div> <div># 4. SMOTE로 샘플링 (소수 클래스 보완)</div> <div>smote = SMOTE(random_state=42)</div> <div>X_resampled, y_resampled = smote.fit_resample(X_train, y_train)</div> <div>from collections import Counter</div> <div>print(Counter(y_train))</div> <div>print(Counter(y_resampled))</div> <div>Counter({np.float32(0.0): 131, np.float32(1.0): 111})</div> <div>Counter({np.float32(1.0): 131, np.float32(0.0): 131})</div>			
				용어설명
내레이션	21			

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	6	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤ 학습하기</div> <div>1. DNN 학습 최적화</div> <div>2. DNN 구조 최적화</div> <div>3. Dataset 최적화</div> <div>➤적용하기</div> <div>➤Outro</div> <div>•문제풀기</div>	<div>• 데이터 증강(Data Augmentation)</div> <div>작은 데이터로 큰 성능을 뽑아내는 핵심 전략</div> <div>모델 학습을 위해 기존 데이터를 변형하거나 추가 데이터를 만들어 더 크고 다양한 학습 세트를 구성하는 기법</div> <div>목표</div> <div>일반화 성능 향상, 과적합 방지</div> <div>방법</div> <div>노이즈 추가</div> <div>각 특징(feature)에 랜덤 정규분포 값 더함</div> <div>회전, 확대, 마스킹, 샘플 재조합 등</div>			
				용어설명

내레이션

22

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	6	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤ 학습하기</div> <div>1. DNN 학습 최적화</div> <div>2. DNN 구조 최적화</div> <div>3. Dataset 최적화</div> <div>➤적용하기</div> <div>➤Outro</div> <div>•문제풀기</div>	<div>• 데이터 증강을 위한 클래스 MixedAugmentedDataset</div> <div>혼합 증강: 원본 데이터 + 데이터 변형(노이즈 데이터)</div> <div>증강 강도 조절 파라미터: 변형된 원본 데이터에 약간의 노이즈 추가(곱)로 다양성 확보</div> <div>noise_std: 노이즈 표준편차, 0.01~0.05</div> <div># 4. 증강 클래스 (원본 데이터 + 노이즈 데이터) 정의</div> <pre>class MixedAugmentedDataset(Dataset):     def __init__(self, X, y, noise_std=0.03, num_augments=1):         ...      def _combine_original_and_augmented(self):         X_list = [self.X[i] for i in range(len(self.X))] # 원본 포함         y_list = [self.y[i] for i in range(len(self.y))]          for i in range(len(self.X)):             for _ in range(self.num_augments):                 noisy = self.X[i] + torch.randn_like(self.X[i]) * self.noise_std                 X_list.append(noisy)                 y_list.append(self.y[i])          return X_list, y_list</pre>			
				용어설명

내레이션

...

23

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	6	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤학습하기</div> <div>1. DNN 학습 최적화</div> <div>2. DNN 구조 최적화</div> <div>3. Dataset 최적화</div> <div>➤적용하기</div> <div>➤Outro</div> <div>•문제풀기</div>	<div>• 데이터 증강(data augmentation) + 샘플링(oversampling/undersampling)</div> <div>동시에 사용할 경우 순서는 매우 중요</div> <div>잘못된 순서로 하면 증강 데이터에 치우친 학습, 샘플링 효과 상실, 또는 데이터 중복 같은 문제가 발생 가능</div> <div>정답은 상황에 따라 다르지만</div> <div>실전에서 추천되는 일반적인 순서는 "샘플링 → 증강"</div> <div>샘플링(특히 SMOTE)은 소수 클래스의 수적 불균형을 먼저 해결</div> <div>그 다음 증강을 해야 균형 잡힌 데이터를 기반으로 더 다양한 패턴을 만들 수 있음</div>		용어설명	
내레이션	<div>24</div>			

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	6	화면설명															
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤ 학습하기</div> <div>1. DNN 학습 최적화</div> <div>2. DNN 구조 최적화</div> <div>3. Dataset 최적화</div> <div>➤ 적용하기</div> <div>➤ Outro</div> <div>•문제풀기</div>		<div>• 데이터 증강 + 샘플링(oversampling/undersampling) 상황별 전략</div> <table><tr><th>상황</th><th>순서</th><th>이유 요약</th></tr><tr><td>클래스 불균형이 심함</td><td>샘플링 → 증강</td><td>먼저 균형 맞추고 다양화</td></tr><tr><td>샘플이 매우 적은 클래스가 있음</td><td>샘플링 → 증강</td><td>부족한 수 보완하고, 다양화</td></tr><tr><td>데이터가 고차원 이미지, 시계열</td><td>조건부</td><td>일부 시계열은 증강 먼저도 가능</td></tr><tr><td>학습 시간/자원 제한</td><td>증강만 단독 사용</td><td>증강만으로도 일정 성능 확보 가능</td></tr></table>			상황	순서	이유 요약	클래스 불균형이 심함	샘플링 → 증강	먼저 균형 맞추고 다양화	샘플이 매우 적은 클래스가 있음	샘플링 → 증강	부족한 수 보완하고, 다양화	데이터가 고차원 이미지, 시계열	조건부	일부 시계열은 증강 먼저도 가능	학습 시간/자원 제한	증강만 단독 사용	증강만으로도 일정 성능 확보 가능	
					상황	순서	이유 요약													
					클래스 불균형이 심함	샘플링 → 증강	먼저 균형 맞추고 다양화													
					샘플이 매우 적은 클래스가 있음	샘플링 → 증강	부족한 수 보완하고, 다양화													
					데이터가 고차원 이미지, 시계열	조건부	일부 시계열은 증강 먼저도 가능													
학습 시간/자원 제한	증강만 단독 사용	증강만으로도 일정 성능 확보 가능																		
		용어설명																		
내레이션					25															

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	6	화면설명																		
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤학습하기</div> <div>1. DNN 학습 최적화</div> <div>2. DNN 구조 최적화</div> <div>3. Dataset 최적화</div> <div>➤적용하기</div> <div>➤Outro</div> <div>•문제풀기</div>	<div>• 심장병 판별 데이터에서 고찰</div> <div>다음 3가지를 적당히 조합하는 것이 가장 좋은 전략</div> <div>소수 클래스 보정(SMOTE) +</div> <div>배치 정규화(BatchNorm, Dropout) +</div> <div>원본 데이터 + 노이즈 추가: MixedAugmentation</div> <table><thead><tr><th>실험 조건</th><th>Accuracy</th><th>F1 Score</th></tr></thead><tbody><tr><td>원본만 학습</td><td>0.82</td><td>0.76</td></tr><tr><td>SMOTE만 적용</td><td>0.85</td><td>0.81</td></tr><tr><td>노이즈만 적용 (std=0.03)</td><td>0.86</td><td>0.82</td></tr><tr><td>SMOTE + 노이즈 (std=0.03)</td><td>0.88</td><td>0.85 ✓</td></tr><tr><td>SMOTE + 노이즈 + Dropout/BN 사용</td><td>0.89</td><td>0.87 ✓✓</td></tr></tbody></table>				실험 조건	Accuracy	F1 Score	원본만 학습	0.82	0.76	SMOTE만 적용	0.85	0.81	노이즈만 적용 (std=0.03)	0.86	0.82	SMOTE + 노이즈 (std=0.03)	0.88	0.85 ✓	SMOTE + 노이즈 + Dropout/BN 사용	0.89	0.87 ✓✓	
	실험 조건	Accuracy	F1 Score																				
	원본만 학습	0.82	0.76																				
	SMOTE만 적용	0.85	0.81																				
	노이즈만 적용 (std=0.03)	0.86	0.82																				
SMOTE + 노이즈 (std=0.03)	0.88	0.85 ✓																					
SMOTE + 노이즈 + Dropout/BN 사용	0.89	0.87 ✓✓																					
					용어설명																		
내레이션	26																						



과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	6	화면설명
<div>➤Intro<ul style="list-style-type: none"><li>•학습열기</li><li>•학습목표</li></ul></div> <div>➤ 학습하기<ul style="list-style-type: none"><li>1. DNN 학습 최적화</li><li>2. DNN 구조 최적화</li><li>3. Dataset 최적화</li></ul></div> <div>➤ 적용하기</div> <div>➤ Outro<ul style="list-style-type: none"><li>•문제풀기</li></ul></div>	<div>• 심장병 판별 결과</div> <div>정확도와 조화 평균이 .9가 넘는 경우</div> <div>Epoch 10   Loss: 0.2971</div> <div>Epoch 20   Loss: 0.2087</div> <div>Epoch 30   Loss: 0.1926</div> <div>Epoch 40   Loss: 0.1723</div> <div>Epoch 50   Loss: 0.1463</div> <div>=== 6강 Evaluation Metrics ===</div> <div>Accuracy : 0.9016</div> <div>Precision: 0.8438</div> <div>Recall : 0.9643</div> <div>F1 Score : 0.9000</div>			
				용어설명

내레이션

27

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	6	화면설명
➤Intro	•학습열기	<ul style="list-style-type: none"><li>딥러닝 모델이 훈련 데이터에 과적합(overfitting)되는 현상을 줄이기 위한 전략을 설계해 보세요. 학습률, 정규화 기법(BatchNorm, Dropout), 데이터 증강, 샘플링 등 이번 회차에서 배운 개념을 기반으로 다양한 방법을 제시하고, 이를 심장병 판별 모델에 어떻게 적용할 수 있을지 설명해 보세요.</li></ul>			① 학습 내용과 관련하여 실제 적용력을 높일 수 있는 문제, 혹은 주제를 작성해 주세요.
	•학습목표				② ex. 사례 제시 후 전문가 의견, 실습과제, 응용 예시 시뮬레이션 등
	➤ 학습하기				③ 저작권 침해가 되지 않도록 내용을 구성해 주세요.
1. DNN 학습 최적화		과적합을 방지하기 위해서는 다음과 같은 전략들을 조합해야 합니다.			④ 출처가 있을 경우 반드시 작성해 주세요.
2. DNN 구조 최적화					
3. Dataset 최적화					
➤적용하기		학습률 스케줄링			
➤Outro		학습 초반에는 높은 학습률로 빠르게 수렴하고, 후반에는 학습률을 낮춰 세밀한 조정을 하도록 스케줄러를 적용합니다. 예를 들어 StepLR이나 CosineAnnealingLR을 사용할 수 있습니다.			
		정규화 기법 활용			
		각 층의 입력 분포를 일정하게 유지하기 위해 BatchNorm1d를 적용하고, 일부 뉴런을 확률적으로 비활성화하여 일반화를 유도하는 Dropout을 함께 사용합니다.			
•문제풀기		데이터 불균형 해소			
		SMOTE와 같은 오버샘플링 기법을 통해 클래스 간 균형을 맞추어 모델이 특정 클래스에 편향되지 않도록 합니다.			
		데이터 다양성 확보			
내레이션		노이즈를 추가하거나 입력 데이터를 살짝 변형하는 Data Augmentation을 통해 모델이 다양한 입력 상황에 대응할 수 있도록 합니다.			