- Read
- Write
- Glob
- Grep

DSPy Bootstrap Few-Shot Optimizer

Goal

Automatically generate and select optimal few-shot demonstrations for your DSPy program using a teacher model.

When to Use

- You have **10-50 labeled examples**
- Manual example selection is tedious or suboptimal
- You want demonstrations with reasoning traces
- Quick optimization without extensive compute

Inputs

Input	Type	Description
`program`	`dspy.Module`	Your DSPy program to optimize
`trainset`	`list[dspy.Example]`	Training examples
`metric`	`callable`	Evaluation function
`max_bootstrapped_demos`	`int`	Max teacher-generated demos (default: 4)
`max_labeled_demos`	`int`	Max direct labeled demos (default: 16)

Outputs

Output	Type	Description
`compiled_program`	`dspy.Module`	Optimized program with demos

Workflow

Phase 1: Setup

```
import dspy
from dspy.teleprompt import BootstrapFewShot

# Configure LMs
dspy.configure(lm=dspy.LM("openai/gpt-4o-mini"))
```

Phase 2: Define Program and Metric

```
class QA(dspy.Module):
    def __init__(self):
        self.generate = dspy.ChainOfThought("question -> answer")

    def forward(self, question):
        return self.generate(question=question)

def validate_answer(example, pred, trace=None):
    return example.answer.lower() in pred.answer.lower()
```

Phase 3: Compile

```
optimizer = BootstrapFewShot(
    metric=validate_answer,
    max_bootstrapped_demos=4,
    max_labeled_demos=4,
    teacher_settings={'lm': dspy.LM("openai/gpt-4o")}
)

compiled_qa = optimizer.compile(QA(), trainset=trainset)
```

Phase 4: Use and Save

```
# Use optimized program
result = compiled_qa(question="What is photosynthesis?")

# Save for production
compiled_qa.save("qa_optimized.json")
```

Production Example

```python
import dspy
from dspy.teleprompt import BootstrapFewShot
from dspy.evaluate import Evaluate
import logging

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

class ProductionQA(dspy.Module):
    def __init__(self):
        self.cot = dspy.ChainOfThought("question -> answer")

    def forward(self, question: str):
        try:
            return self.cot(question=question)
        except Exception as e:
            logger.error(f"Generation failed: {e}")
            return dspy.Prediction(answer="Unable to answer")

def robust_metric(example, pred, trace=None):
    if not pred.answer or pred.answer == "Unable to answer":
        return 0.0
    return float(example.answer.lower() in pred.answer.lower())

def optimize_with_bootstrap(trainset, devset):
    """Full optimization pipeline with validation."""

    # Baseline
    baseline = ProductionQA()
    evaluator = Evaluate(devset=devset, metric=robust_metric, num_threads=4)
    baseline_score = evaluator(baseline)
    logger.info(f"Baseline: {baseline_score:.2%}")

    # Optimize
    optimizer = BootstrapFewShot(
        metric=robust_metric,
        max_bootstrapped_demos=4,
        max_labeled_demos=4
    )

    compiled = optimizer.compile(baseline, trainset=trainset)
    optimized_score = evaluator(compiled)
    logger.info(f"Optimized: {optimized_score:.2%}")

    if optimized_score > baseline_score:
        compiled.save("production_qa.json")
        return compiled

    logger.warning("Optimization didn't improve; keeping baseline")
    return baseline
```

Best Practices

1. **Quality over quantity** - 10 excellent examples beat 100 noisy ones
2. **Use stronger teacher** - GPT-4 as teacher for GPT-3.5 student
3. **Validate with held-out set** - Always test on unseen data

4. **Start with 4 demos** - More isn’t always better

Limitations

- Requires labeled training data
- Teacher model costs can add up
- May not generalize to very different inputs
- Limited exploration compared to MIPROv2