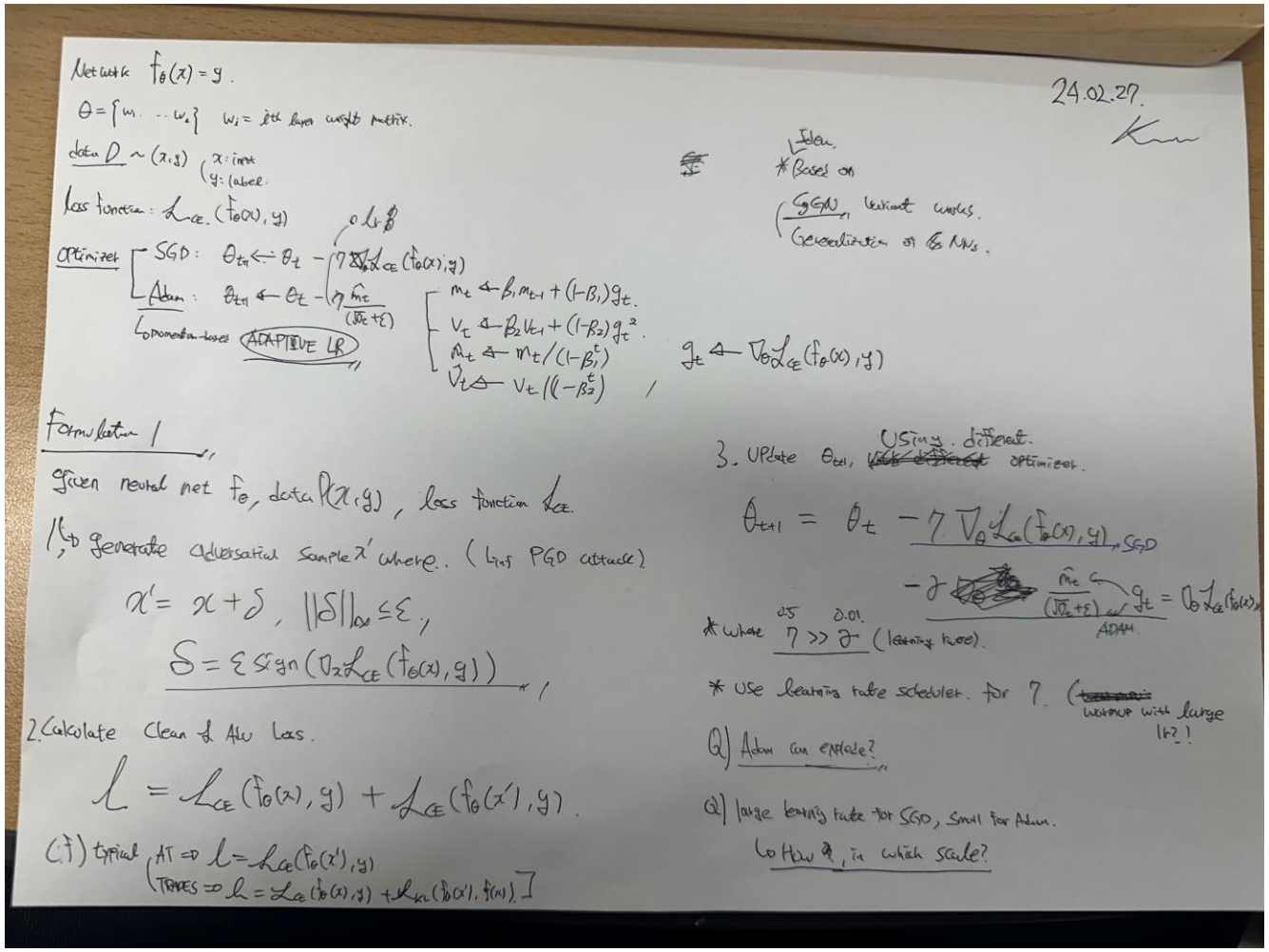
Formulation

KwanHee Lee
Graduation Project
CSE@Konkuk Univ.
24.02.21

Contents

1. Detailed formulation
2. Code Implementation
3. Disuccusion

Detailed formulation



Code Implementation

Optimization process

$$\theta_{t+1} \leftarrow \theta_t - \eta \nabla_\theta \mathcal{L}_{ce}(f_\theta(x), y) - \gamma \nabla_\theta \frac{m}{\sqrt{v_t} + \epsilon} \mathcal{L}_{ce}(f_\theta(x_{adv}), y)$$

Implementation using torch.autograd.grad()

```
loss_natural.backward() # maintains loss in computational graph
adv_gradients = torch.autograd.grad(loss_robust,model.parameters()) # gradients w.r.t adv loss
optimizer_sgd.step() # update natural loss

optimizer_adam.zero_grad()    # zero gradient
for param,grad in zip(model.parameters(),adv_gradients): # update param.grad with previous gradients w.r.t adv loss
    param.grad = grad
optimizer_adam.step() # update adversarial loss

loss = loss_natural + beta*loss_robust # meaningless
```

- Autograd.grad() -> calculates gradient and return tensor(gradient)
- backward() -> calculates gradient and accumulates it into tensor.grad
- Optimizer.step() -> updates parameter w.r.t tensor.grad using optimizer
- Optimizer.zero_grad() -> removes tensor.grad w.r.t given parameter

Discussion

- Need to specify hyperparameters..
- Will it work out?