Continual Learning

Regularization Priors

Antonio Carta

antonio.carta@unipi.it

Outline

- CL regularization as **past loss approximation**
- **EWC**: an information-geometric approximation of past task loss

- We will also discuss **Natural Gradient Descent**
  - This is unrelated to our goal of mitigating forgetting but it's a popular application of the same theory behind EWC

Proxy Losses

Tools to find a good surrogate loss for the old data

Continual Learning Objective

CL Loss:

$$\mathcal{L}^{\text{tot}}(\theta) = \mathcal{L}^{\text{new}}(\theta) + \mathcal{L}^{\text{old}}(\theta)$$

- We estimate $\mathcal{L}^{\text{new}}(\theta)$ from new data (easy)
- How do we estimate $\mathcal{L}^{\text{old}}(\theta)$?
  - We don't have access to the old data
  - We need to approximate it

Bayesian vs Optimization Approach

- Given the current model θ_t trained up to experience t
- **Bayesian** approach:
  - We have a posterior $p_t(\theta)$
  - Solution: compute new posterior $p_{t+1}(\theta)$ usando $p_t(\theta)$ come prior (bayesian sequential update)
  - Approximation: the posterior will be approximated, which will introduce errors over time
- **Optimization** approach:
  - Find a suitable approximation $\tilde{\mathcal{L}}^{old}(\theta)$ for $\mathcal{L}^{\text{old}}(\theta)$
  - Optimize $\tilde{\mathcal{L}}^{old}(\theta) + \mathcal{L}^{\text{new}}(\theta)$
  - The approximation will use local information about loss at the current solution θ_t, which is a minima for old experiences

The Optimization Approach

- We can use Taylor to approximate $\mathcal{L}^{\text{old}}$ around the previous solution θ_t

$$\mathcal{L}^{\text{old}}(\theta) \approx \mathcal{L}^{\text{old}}(\theta_t) + \nabla\mathcal{L}^{\text{old}}(\theta_t)^T(\theta - \theta_t) + \frac{1}{2}(\theta - \theta_t)^T Hf(\theta_t)(\theta - \theta_t)$$

- This approximation provides a surrogate loss that we can estimate without the old data
- The loss for the old experiences with parameters θ is approximated with a quadratic loss function

The Optimization Approach

$$\mathcal{L}^{\text{old}}(\theta) \approx \mathcal{L}^{\text{old}}(\theta_t) + \nabla\mathcal{L}^{\text{old}}(\theta_t)(\theta - \theta_t) + \frac{1}{2}(\theta - \theta_t)^T Hf(\theta_t)(\theta - \theta_t)$$

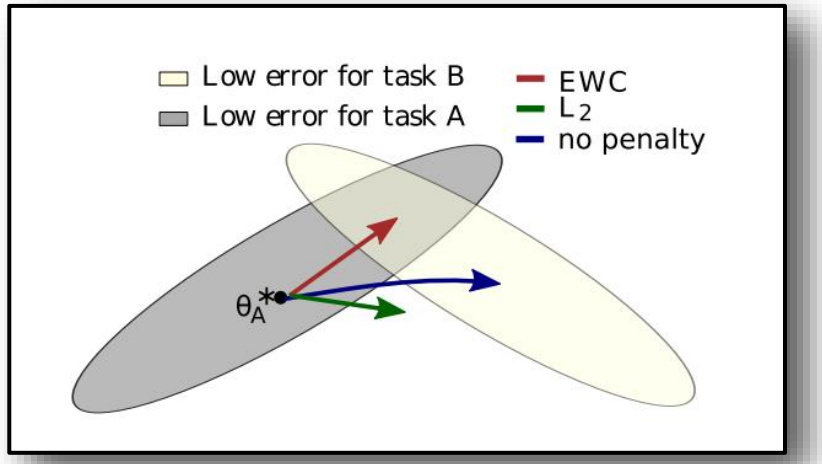
When we minimize the surrogate loss:

- θ_t is the previous solution and it's constant. $\mathcal{L}^{\text{old}}(\theta_t)$ is a constant that we can ignore
- θ_t is a minimum of $\mathcal{L}^{\text{old}}(\theta_t)$. Therefore $\nabla\mathcal{L}^{\text{old}}(\theta_t) \approx 0$, $\nabla\mathcal{L}^{\text{old}}(\theta_t)^T(\theta - \theta_t) \approx 0$ and we can ignore it
- We will use a diagonal approximation of $Hf(\theta_t)$

The result is a loss of the form $\sum_i h_i\,(\theta^i - \theta_t^i)^2$

Now we «only» need to find a good approximation for the (diagonal) of the hessian.

Regularization as Importance



- Our loss is $\sum_i h_i \, (\theta^i - \theta_t^i)^2$
- This is a weight decay
  - Centered around the previous solution
  - Non-uniform
- The coefficients h_i are the curvature of the loss around θ_t
- Informally, we can think of h_i as the importance of parameter i
- Keep in mind that this is only a local approximation

Forgetting and Parameter Importance

DNNs are overparameterized:

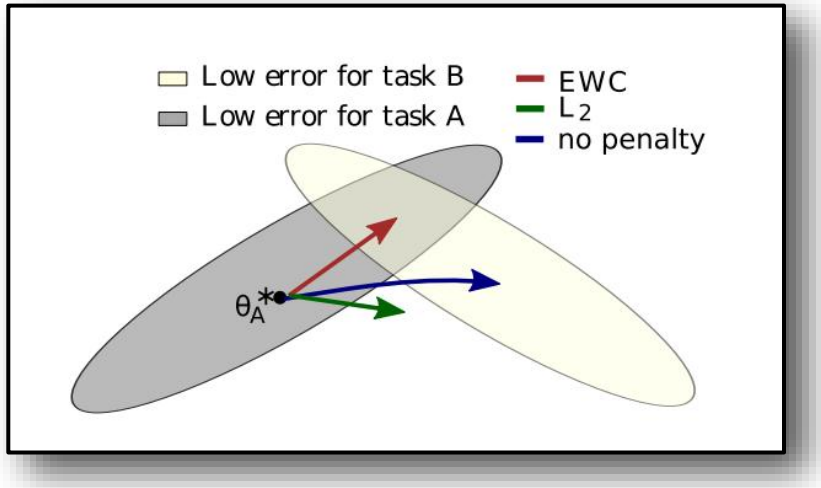
- For each task, we expect that only few parameters are important (high h_i)
- The loss penalizes changes in the important parameters (high curvature) while leaving the unimportant ones free to adapt to the new task

$$\sum_i h_i \left(\theta^i - \theta_t^i \right)^2$$

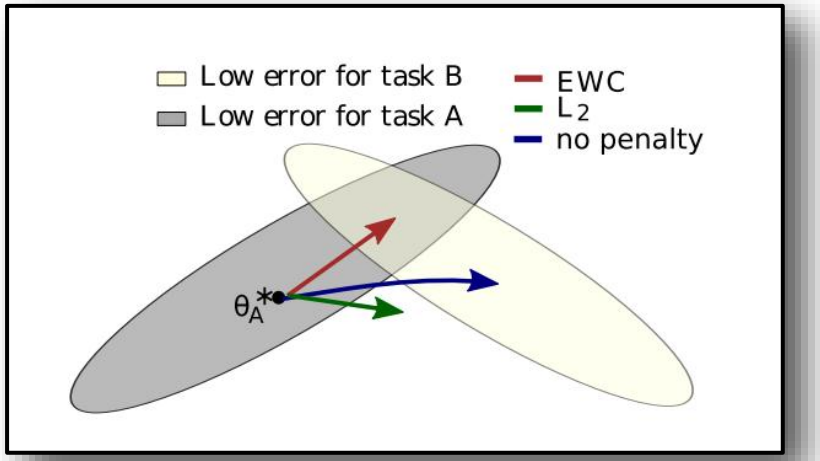
Regularization as Distance between Solutions

- Can we measure the distance between the old and new solution?
- What is a good distance measure?

$$L(\theta) = L^{new}(\theta) + D\left(\theta, \theta^{old}\right)$$



Bayesian Sequential Udpate



- Can we compute the posterior?
  - Not really with DNN…
- Can we at least approximate it?
  - Yes, with some «brutal» approximations
  - The approximation will cause some forgetting over time
  - It may still be the best that we can do
- If we have an approximate posterior, we can use a sequential bayesian update:

$$\log p(\theta \mid \mathcal{D}) = \log p(\mathcal{D}_{new} \mid \theta) + \log p(\theta \mid \mathcal{D}_{old}) - \log p(\mathcal{D}_{new})$$

Recap

- Our goal is to find a good approximation of loss for the past data
- In order to do that we need some information, such as
  - The curvature of the loss and a good minima
  - A distance metric between solution
  - An importance measure for the model's parameters
  - The posterior of the weights
- **PROBLEM**: We still need to find an effective approximation for these quantities

Fisher Information

Measuring distances in the probability space

Distance between solutions

- We can think of our loss $\sum_i h_i \, (\theta^i - \theta_t^i)^2$ as a distance measure between two models: θ and θ_t

Is it a good distance measure?

- Yes, if h_i are a good approximation of the curvature of the loss
- Can we find other good importance measures that are easy to compute?
- Can we use the l2 distance $\sum_i (\theta^i - \theta_t^i)^2$?
  - In principle, yes, it is a good heuristic if for some reason we cannot estimate parameter importance
- **IDEA**: we don't really care about the parameters, we care about the distance between the probability distributions $f(.,;\theta)$ and $f(.;\theta_t)$

Kullback Leibler Divergence

- KL divergence measures the distance between two probability distributions

$$D_{\mathrm{KL}}(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) \mathrm{d}x$$

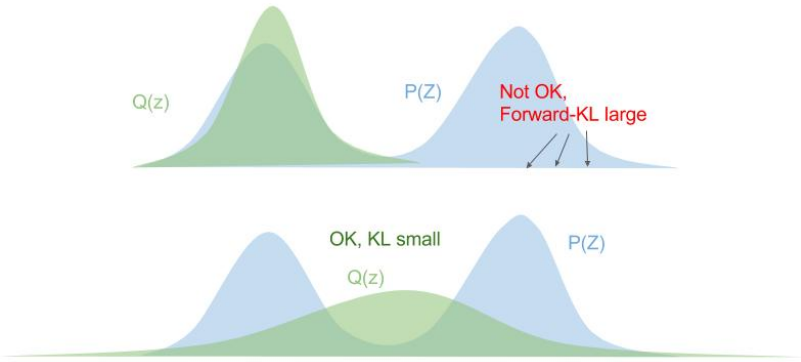
- It can be computed in closed-form for simple distributions
- Otherwise, we can estimate it using the data
- 1D gaussian: $KL(\mu_1, \sigma_1, \mu_2, \sigma_2) = \log\frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}$

Side note: KL is not symmetric

- $D_{\mathrm{KL}}(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) \mathrm{d}x$
- In the examples below P is a bimodal distribution, while Q is unimodal
- We minimize the forward and reverse KL by fixing P and optimizing Q

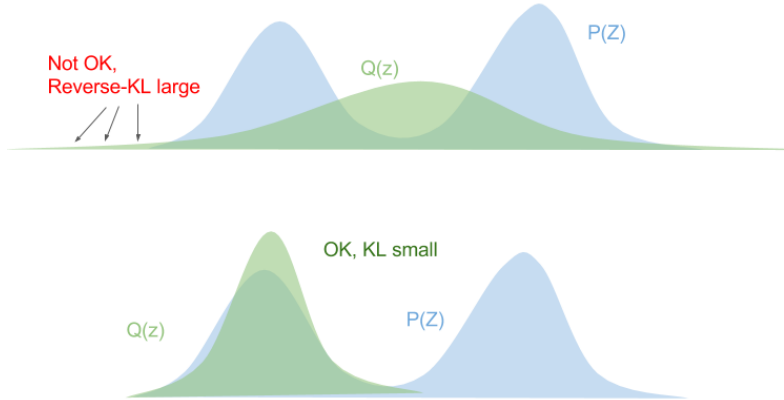
«forward KL» KL(P || Q)

- Q will take the average mode
- Q(Z) is known as zero-avoiding
- minimizing forward-KL stretches your variational distribution Q(Z) to cover over the entire P(Z)



«reverse KL» KL(Q || P)

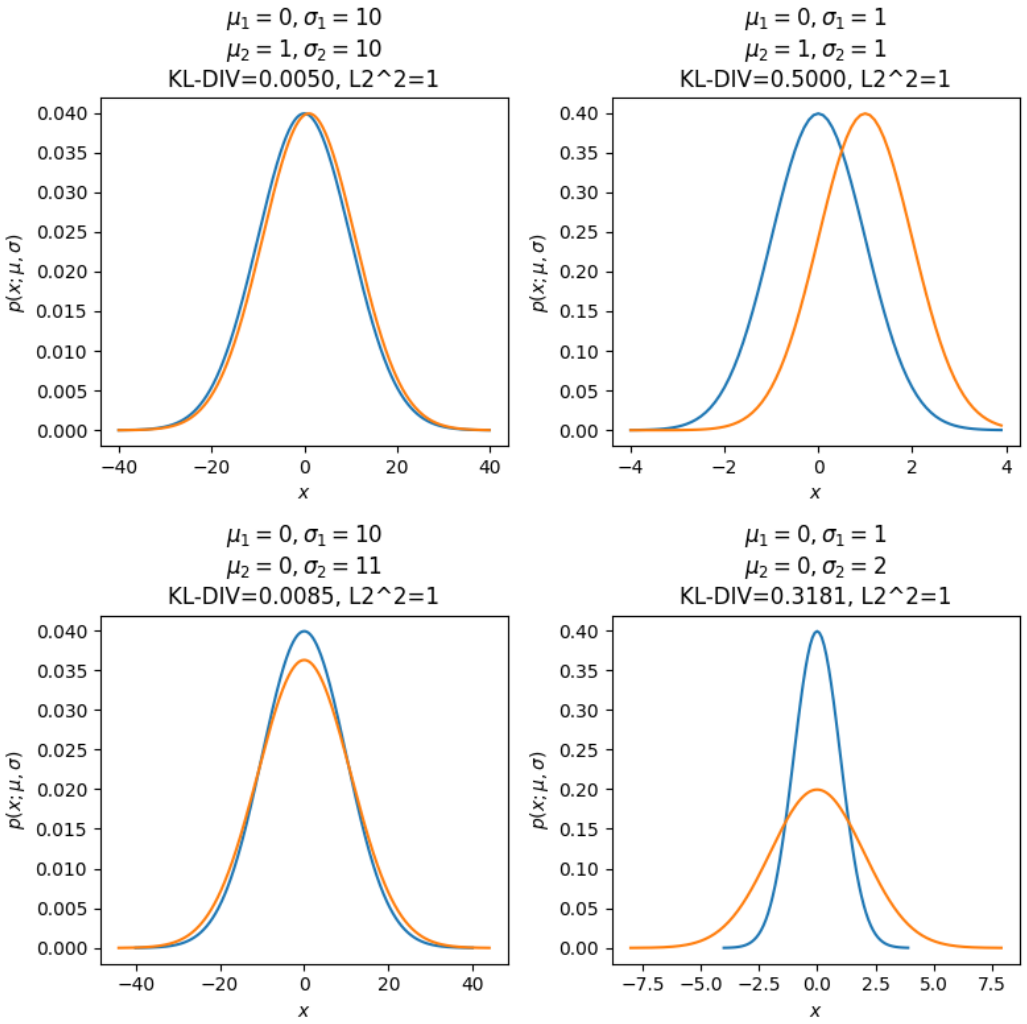
- Q will take a single mode
- If p(Z)=0, we must ensure that the weighting function q(Z)=0 wherever denominator p(Z)=0, otherwise the KL blows up.
- This is known as "zero-forcing«
- reverse-KL "squeezes" the Q(Z) under P(Z).



Images from [Variational Bayes and The Mean-Field Approximation | Bounded Rationality (bjlkeng.io)]

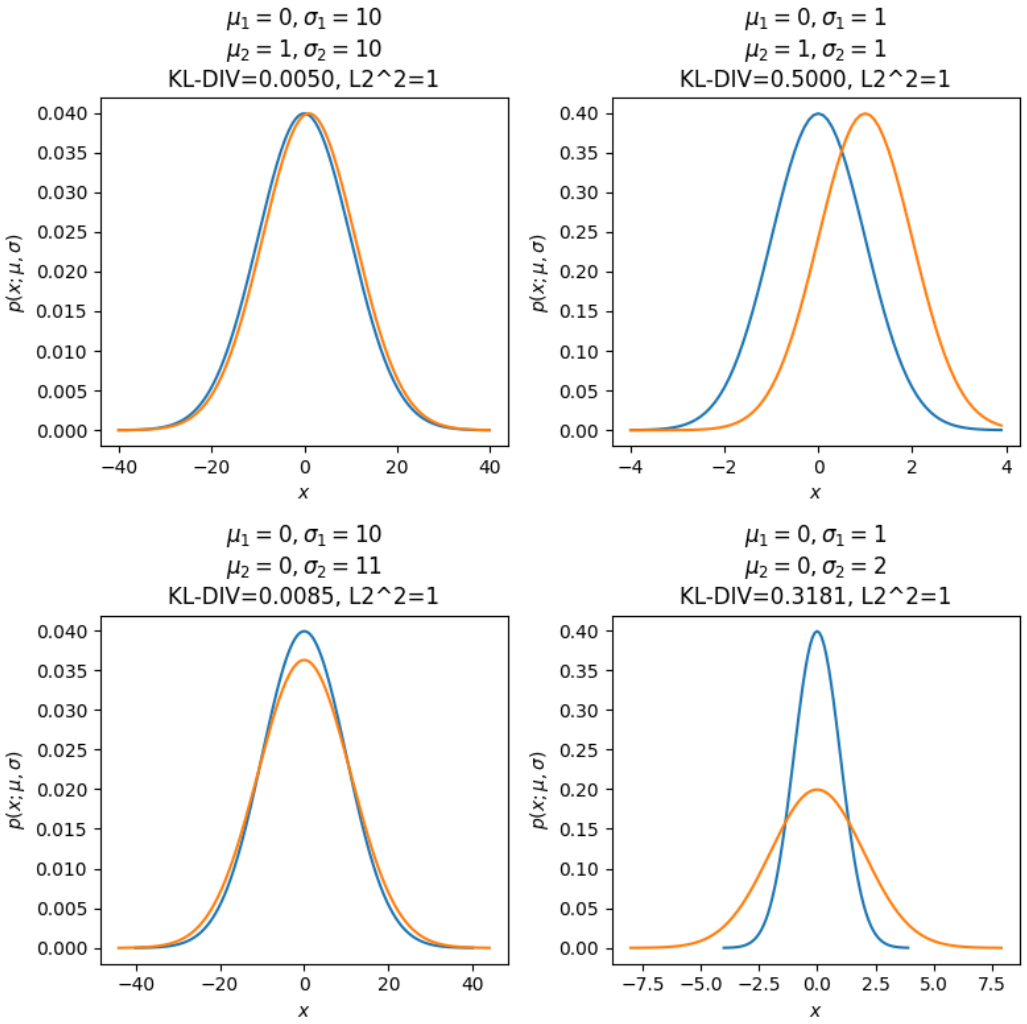
Example – 1D Gaussians

- All the plots have the same l2 distance in the parameter space (=1) but are very different in the «function space»
- The KL-divergence computes the distance between the two distributions
- 1D gaussian: $KL(\mu_1, \sigma_1, \mu_2, \sigma_2) = \log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}$



Example – 1D Gaussians

- If we make a small change to the blue distribution, in the top-right picture we get a much bigger change in KL distance
- We want to:
  - take into account the «local geometry» during SGD
  - Try to change only the parameters that won't increase the KL distance wrt the previous model too much



Fisher Information

- **Fisher Information**

$$\mathrm{F}(\theta^*) = \mathrm{E}\left[\left(\frac{\partial}{\partial\theta}\ln f(X;\theta^*)\right)^2\right]$$

- The Fisher information is equivalent to the curvature of the $KL(\theta\,|\theta^*)$ in θ^*
- The Fisher information $\mathrm{F}(\theta)$ measures how sensitive the distribution is wrt changes to each parameter

Fisher Information

- **Fisher Information**: $\mathrm{F}(\theta^*) = \mathrm{E}\left[\left(\frac{\partial}{\partial\theta}\ln f(X;\theta^*)\right)^2\right]$

- The KL is not the loss function, but it's a good distance measure and often close to the loss function (e.g. crossentropy)
- Easy to compute: we only need the gradients
- In our setting, the expectation is taken by sampling $p(x)$ from the data and $p(y|x)$ from the model (not the data!)

Fisher and KL Divergence

- Taylor expansion of the KL
  - $KL(\theta|\theta^*) \approx KL(\theta^*|\theta^*) + \nabla KL(\theta^*|\theta^*)^T(\theta - \theta^*) + \frac{1}{2}(\theta - \theta^*)^T H_{\theta^*}(\theta - \theta^*)$
- We have
  - $KL(\theta^*|\theta^*) = 0$
  - $\nabla KL(\theta|\theta^*) \quad = 0$
  - Where H_{θ^*} is the Fisher information
- And we are left with
  - $KL(\theta|\theta^*) \approx \frac{1}{2}(\theta - \theta^*)^T H_{\theta^*}(\theta - \theta^*)$
- If we approximate the hessian with a diagonal matrix we end up with
  - $\sum_i h_i (\theta - \theta^*)^2$

How to compute the Fisher Information

- **True Fisher Information**:
  - $\mathrm{F}_{\prod_n p_\theta(x,y)}(\theta) = N\mathbb{E}_{x,y\sim p(x)p_\theta(y|x)}[\nabla_\theta \log p_\theta(y \mid x)\nabla_\theta \log p_\theta(y \mid x)^T]$
  - x sampled from the true distribution, $p_\theta(y \mid x_n)$ is the model's output distribution
  - We don't know $p(x)$ so we can't compute this quantity
- **Fisher Information**:
  - $\mathrm{F}_{\prod_n p_\theta(y|x_n)}(\theta) = \sum_n \mathbb{E}_{y\sim p_\theta(y|x_n)}[\nabla_\theta \log p_\theta(y \mid x_n)\nabla_\theta \log p_\theta(y \mid x_n)^T]$
  - x sampled from the dataset, $p_\theta(y \mid x_n)$ is the model's output distribution
  - we can compute this and it is the quantity most frequently used by method that require the Fisher information
- **Empirical Fisher Information**:
  - $\tilde{\mathrm{F}}(\theta) = \sum_n \nabla_\theta \log p_\theta(y_n \mid x_n)\nabla_\theta \log p_\theta(y_n \mid x_n)^T$
  - Here, x_n, y_n are both sampled from the dataset
  - This is easier to implement but it is not a good approximation of the curvature of the KL (see slide 28 for an example)
  - It is not a Monte Carlo estimate of the Fisher Information

A word of advice: most code on github computes the empirical Fisher (and it is often wrong in other subtle ways)

Kunstner, Frederik, Philipp Hennig, and Lukas Balles. "Limitations of the empirical fisher approximation for natural gradient descent." *Advances in neural information processing systems* 32 (2019).

PyTorch – Fisher Information

```
for x, y, t in dl:
    for p in model.parameters():  # zero-grad
        p.grad = None
    yp = model(x, t)
    yp = yp[0]  # there is a single example in the mini-batch

    # we need to sample from yp = p(y | x)
    p = Categorical(torch.softmax(yp, dim=0))
    ysample = p.sample()

    # log-likelihood of sampled class
    ll = torch.log_softmax(yp, dim=0)[ysample]
    ll.backward()

    # we compute the fisher only for the feature extractor
    for n, p in model.model.features.named_parameters():
        if n in F:
            F[n] += p.grad ** 2
        else:
            F[n] = p.grad ** 2
    cnt += 1

for n, p in F.items():
    F[n] = p / cnt
```

Information Geometry Applications

- The Fisher information provides information about the local geometry of the statistical manifold of probability distributions represented by our model
- We know which parameters are more sensitive to infinitesimal changes (measured by the KL distance)

- We can use this information for
  - Speed up training with **Natural Gradient Descent**
  - Prevent forgetting with **Elastic Weight Consolidation**

Natural Gradient Descent

Fast and stable SGD for statistical manifolds

SGD Distance Metric

- **SGD step**: $\theta_i = \theta_{i-1} - \alpha \nabla L(\theta_{i-1})$
- When we do a descent step, the learning rate is the same for every parameter
  - Basically, we can think of SGD as using the L2 distance to measure distances between parameters
- However, we know that same parameters are more sensitive than others (in the KL distance)
- Can we account for the KL distance when we do a SGD step?

SGD with KL Distance

- We can reframe SGD as a constrained optimization problem:

$$\Delta\theta^* = \arg\min_{\Delta\theta} \mathcal{L}(\theta + \Delta\theta)$$
$$\text{s.t. } D_{KL}(p_\theta \parallel p_{\theta+\Delta\theta}) = c$$

- Now, we are ensuring that SGD steps consider the KL distance
- NOTE: this is equivalent to SGD if we use the l2 distance in the constaint instead of the KL

SGD with KL Distance

- To solve the constrained optimization problem we use the Lagrangian approximating both the loss and the KL with their Taylor expansion:

$$\Delta\theta^* = \arg\min_{\Delta\theta} \mathcal{L}(\theta) + \nabla_\theta \mathcal{L}(\theta)\Delta\theta + \frac{1}{2}\lambda\Delta\theta^\top \mathcal{I}(\theta)\Delta\theta - \lambda c$$

- Remember: the hessian $\mathcal{I}(\theta)$ is the Fisher information!
- By computing the gradient w.r.t. Δθ and setting it to zero:

$$\nabla_\theta \mathcal{L}(\theta) + \lambda\mathcal{I}(\theta)\Delta\theta = 0$$

$$\Delta\theta^* = \frac{1}{\lambda}\mathcal{I}(\theta)^{-1}\nabla_\theta \mathcal{L}(\theta)$$

- We can ignore the factor 1/λ, which will become the learning rate in our method.

Natural Gradient Descent (NGD)

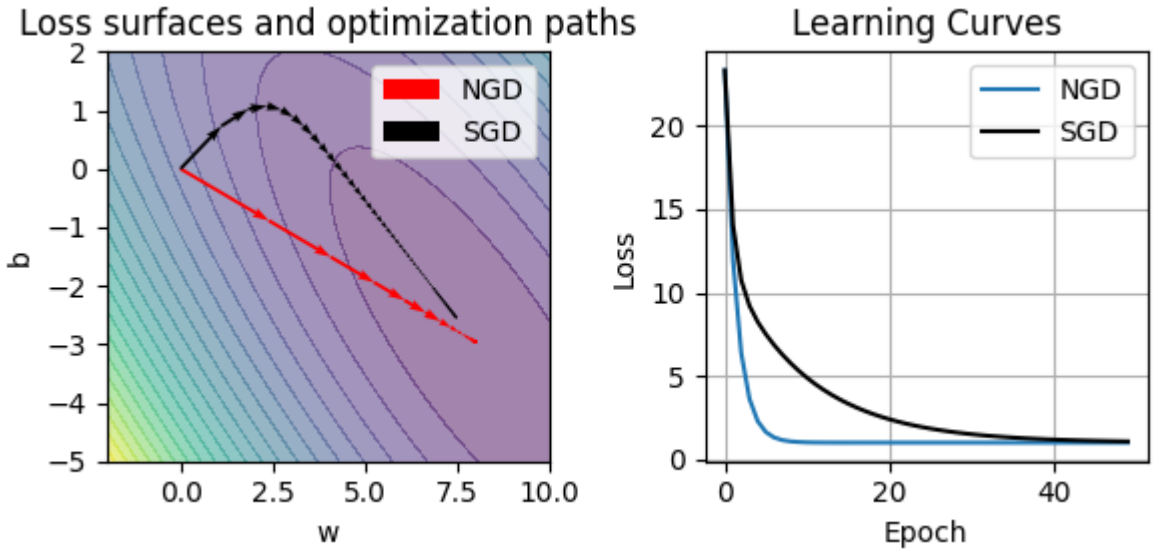
- Natural Gradient Descent update:

$$\theta = \theta - \eta\, F(\theta)^{-1} \nabla_\theta \mathcal{L}(\theta)$$

- η is the learning rate
- $F(\theta)^{-1}$ the inverse of the Fisher Information Matrix
- The update steps considers the local curvature of the KL
- Intuitively, more sensitive parameters are moved less, while less sensitive parameters are moved more (we multiply by the inverse)

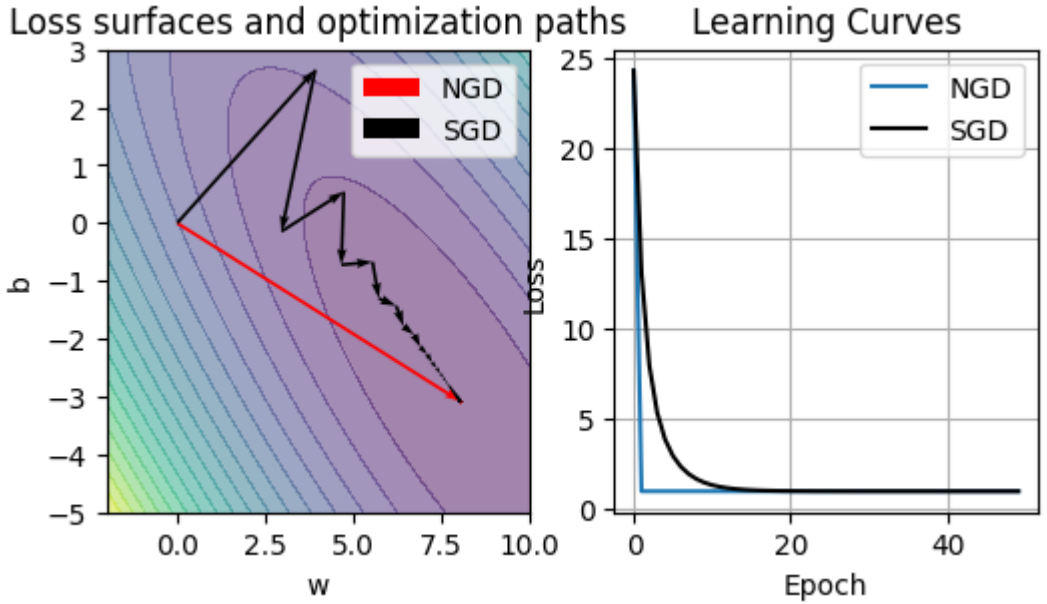
Example: linear regression

- Simple linear regression problem (see notebook for implementation)
- SGD follows the gradient of the loss with a constant learning rate, ignoring the differences between the two parameters (weight and bias)
- NGD converges much faster



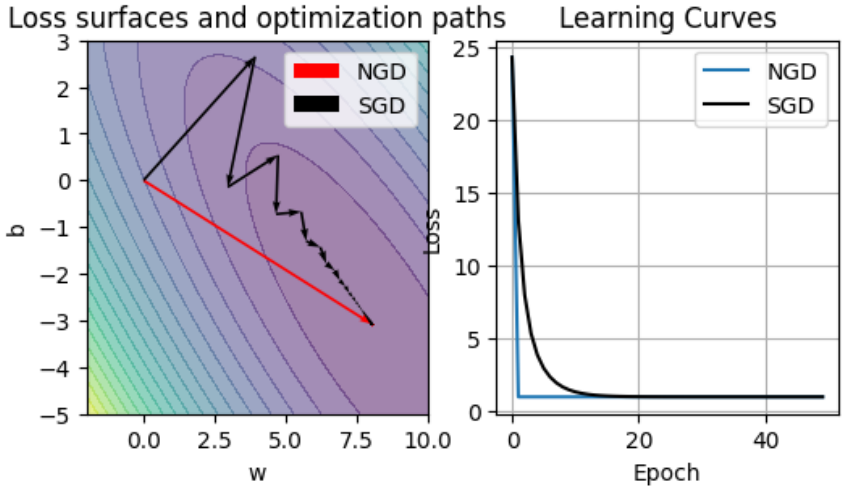
NGD is more stable

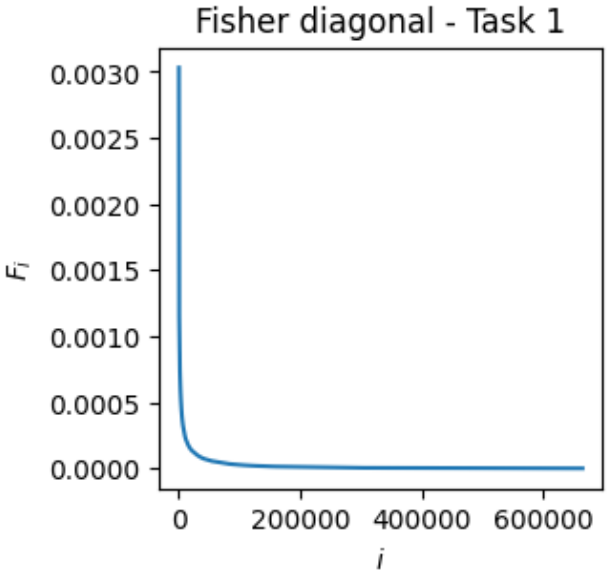
- Compared to the previous slide, we only increased the learning rate from 0.15 to 0.5
- NGD is also more stable



Training Stability

- The previous example may look like a toy example but it's actually much easier than DNN training
- In practice, DNN optimization looks **much worse** than this problem
  - Few parameters have high fisher info
  - Most parameters have a low fisher info
- We can plot the (diagonal of the) Fisher and look at its distribution
  - «flat distribution»: a good parameterization, where naive SGD will be as good as NGD
  - «skewed distribution»: some parameters are more important and SGD will be slower or less stable than NGD depending on the chosen learning rate
- Bottom figure: fisher diagonal after learning on Task 1 of SplitMNIST (see notebook)





Fisher vs Empirical Fisher

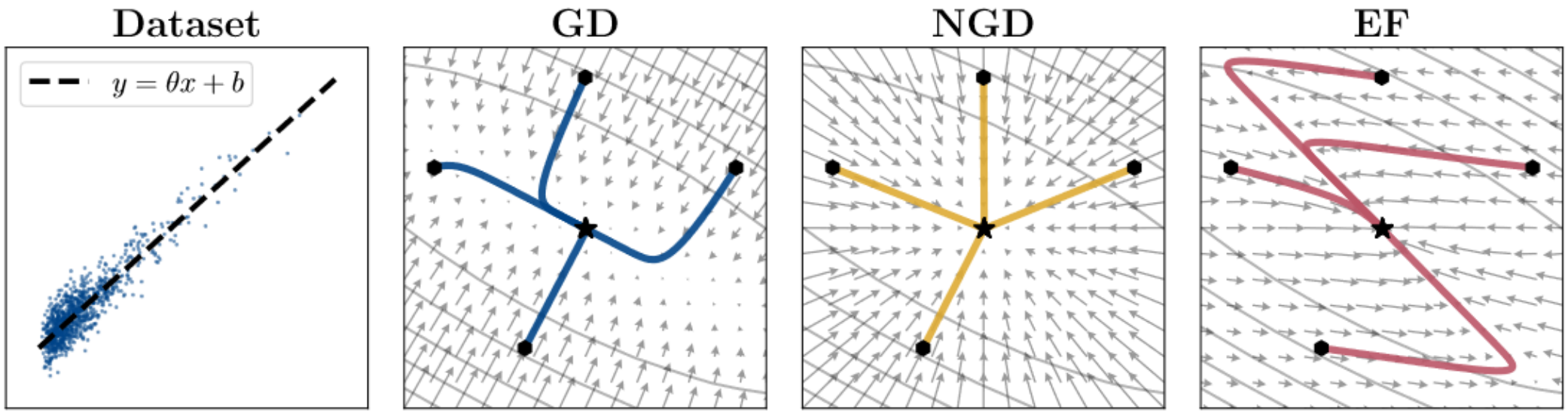


Figure 1: Fisher vs. empirical Fisher as preconditioners for linear least-squares regression on the data shown in the left-most panel. The second plot shows the gradient vector field of the (quadratic) loss function and sample trajectories for gradient descent. The remaining plots depict the vector fields of the natural gradient and the "EF-preconditioned" gradient, respectively. NGD successfully adapts to the curvature whereas preconditioning with the empirical Fisher results in a distorted gradient field.

Kunstner, Frederik, Philipp Hennig, and Lukas Balles. "Limitations of the empirical fisher approximation for natural gradient descent." *Advances in neural information processing systems* 32 (2019).

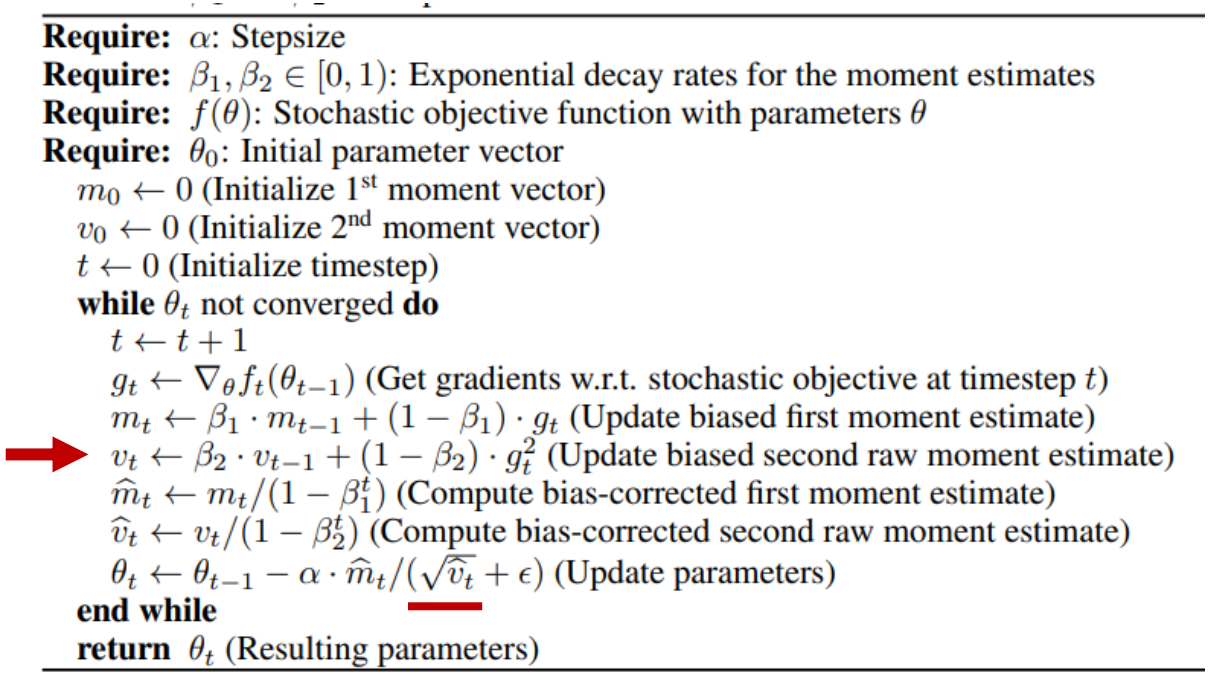
How to compute the Fisher Information

- **True Fisher Information**:
  - $\mathrm{F}_{\prod_n p_\theta(x,y)}(\theta) = N\mathbb{E}_{x,y\sim p(x)p_\theta(y|x)}[\nabla_\theta \log p_\theta(y \mid x)\nabla_\theta \log p_\theta(y \mid x)^T]$
  - x sampled from the true distribution, $p_\theta(y \mid x_n)$ is the model's output distribution
  - We don't know $p(x)$ so we can't compute this quantity
- **Fisher Information**:
  - $\mathrm{F}_{\prod_n p_\theta(y|x_n)}(\theta) = \sum_n \mathbb{E}_{y\sim p_\theta(y|x_n)}[\nabla_\theta \log p_\theta(y \mid x_n)\nabla_\theta \log p_\theta(y \mid x_n)^T]$
  - x sampled from the dataset, $p_\theta(y \mid x_n)$ is the model's output distribution
  - we can compute this and it is the quantity most frequently used by method that require the Fisher information
- **Empirical Fisher Information**:
  - $\tilde{\mathrm{F}}(\theta) = \sum_n \nabla_\theta \log p_\theta(y_n \mid x_n)\nabla_\theta \log p_\theta(y_n \mid x_n)^T$
  - Here, x_n, y_n are both sampled from the dataset
  - This is easier to implement but it is not a good approximation of the curvature of the KL (see slide 28 for an example)
  - It is not a Monte Carlo estimate of the Fisher Information

Kunstner, Frederik, Philipp Hennig, and Lukas Balles. "Limitations of the empirical fisher approximation for natural gradient descent." *Advances in neural information processing systems* 32 (2019).

Adam

- In Adam, $\hat{v}_t$ is an approximation to the diagonal of the Fisher information matrix (Pascanu & Bengio, 2013)
- Adam's preconditioner (like AdaGrad's) is more conservative than NGD by preconditioning with the square root of the inverse



Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).

Recap

- The Fisher Information provide a local distance measure for our model
- We can exploit it to make the gradient descent faster and more stable

NGD and the Fisher information appear everywhere in ML models:

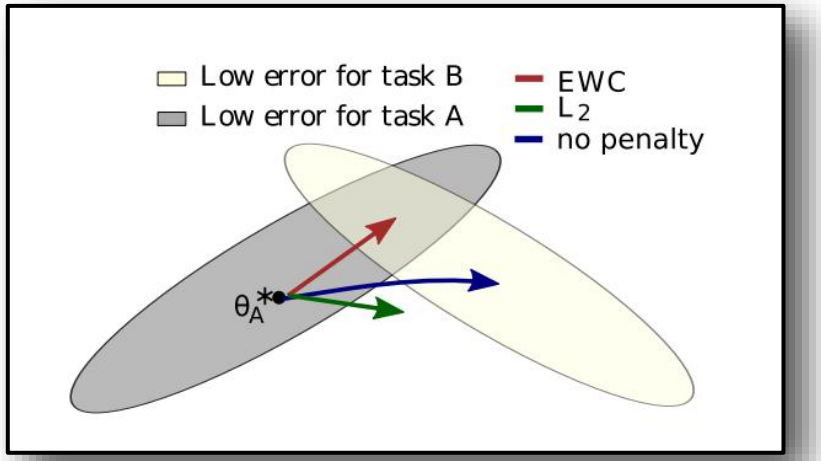
- Many methods employ some form of NGD under the hood
- Others using the natural parameters, a parametrization where we don’t need the Fisher info (unfortunately, we don’t have this luxury with DNNs)

Elastic Weight Consolidation

Using the Fisher Information to prevent forgetting

Importance-Based Methods

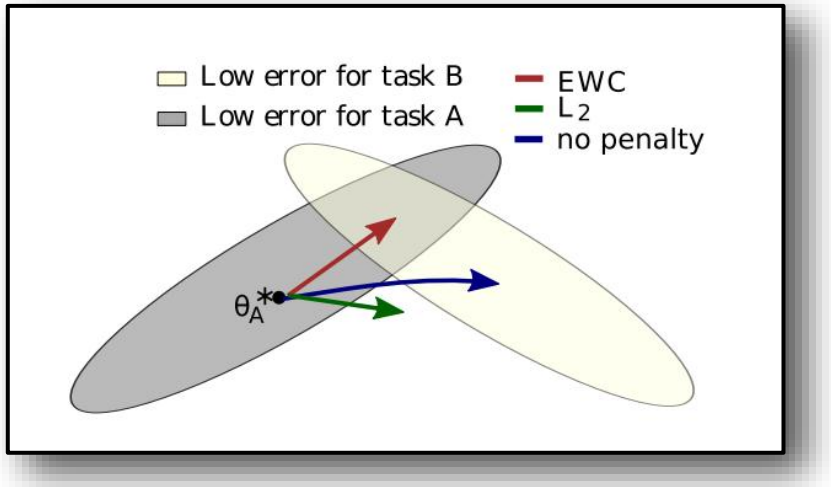
- An example of **prior-focused** method: Model loss function of previous tasks with surrogate losses.
- **IDEA**: important weights should not change. We can change unimportant weights only.
  - Deep networks are overparameterized. This means that we should have a lot of free capacity.
  - Each task should have a small number of important weights.
- **PROBLEM**: how do we find the important weights? Fisher Information!



Overcoming catastrophic forgetting in neural networks, Kirkpatrick et al, PNAS 2017.

EWC – Setting

- Large batches
- We know task boundaries
- We have enough data to find a good model (i.e. the model at the boundaries is optimal for the current task)
- Designed for task labels, but can work without them



Elastic Weights Consolidation (EWC)

Key Idea: after training, store tuple of

$< \theta^{i-1}, F^{i-1} >$

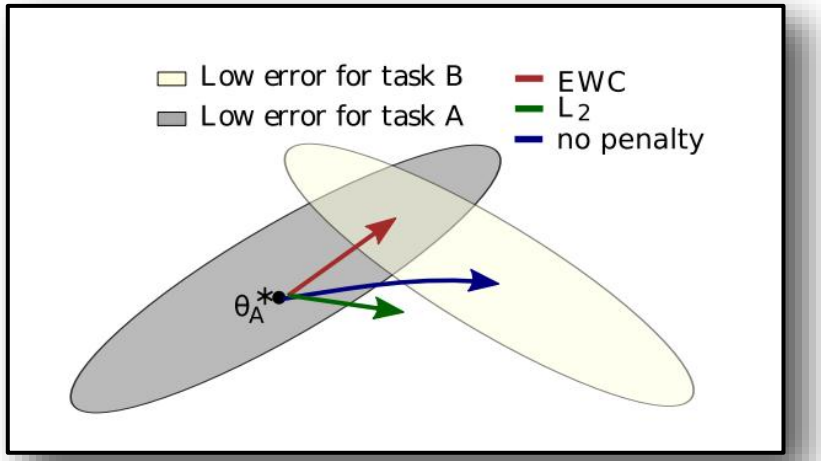
- θ^{i-1} model after training
- F^{i-1} Fisher info at θ^{i-1} estimated with data D_{i-1}

At the end of each experience:

- Store model
- Compute Fisher info

During Training:

- Add EWC loss to the total loss



$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta^*_{A,i})^2$$

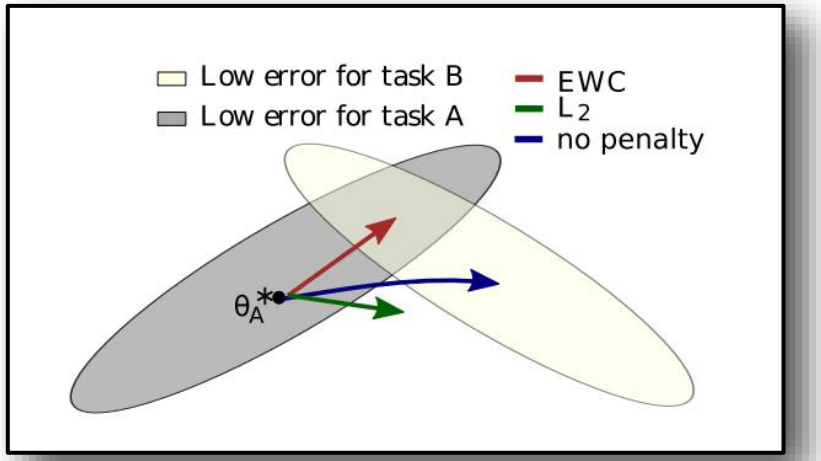
Overcoming catastrophic forgetting in neural networks, Kirkpatrick et al, PNAS 2017.

Elastic Weights Consolidation (EWC)

EWC loss is like the l2 loss with:

- F^{i-1}, the Fisher diagonal coefficients as weights for each parameter
- The previous weights θ^{i-1} as the "center" of the loss (i.e. zero loss point)

Elastic: weights are pulled towards the previous solution (like the l2 decay), weighted by their importance.



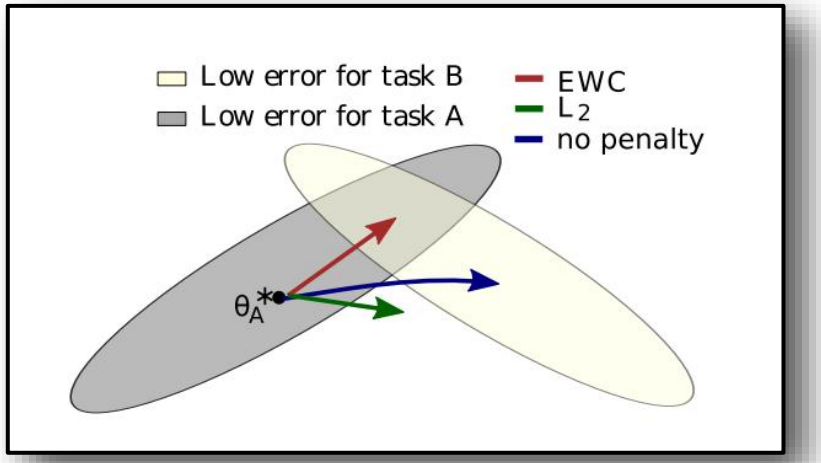
$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta^*_{A,i})^2$$

Overcoming catastrophic forgetting in neural networks, Kirkpatrick et al, PNAS 2017.

Elastic Weights Consolidation (EWC)

How many importance values do you need to keep?

- **separate**: One tuple $< \theta^{i-1}, F^{i-1} >$ for each task.
  - Linear cost, i.e. as much as keeping a separate model for each experience.
- **online**: Online estimate of the Fisher, where the previous Fisher is updated with an average or an exponential moving average(EMA) of the fisher values
  - It corresponds to the sequential Bayesian update
- Very sensitive to regularization strength λ



$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta^*_{A,i})^2$$

Overcoming catastrophic forgetting in neural networks, Kirkpatrick et al, PNAS 2017.

EWC – Laplace Approximation

- We can also think of the Fisher as an approximation of the posterior when using the sequential bayesian update scheme where the posterior becomes the new prior

$$\log p(\theta \mid \mathcal{D}) = \log p(\mathcal{D}_{new} \mid \theta) + \log p(\theta \mid \mathcal{D}_{old}) - \log p(\mathcal{D}_{new})$$

Laplace Approximation:

- True posterior $p(\theta \mid \mathcal{D}_{old})$ is intractable
- We approximate it with a gaussian with mean θ_t (MAP solution) and covariance $F(\theta_t)$

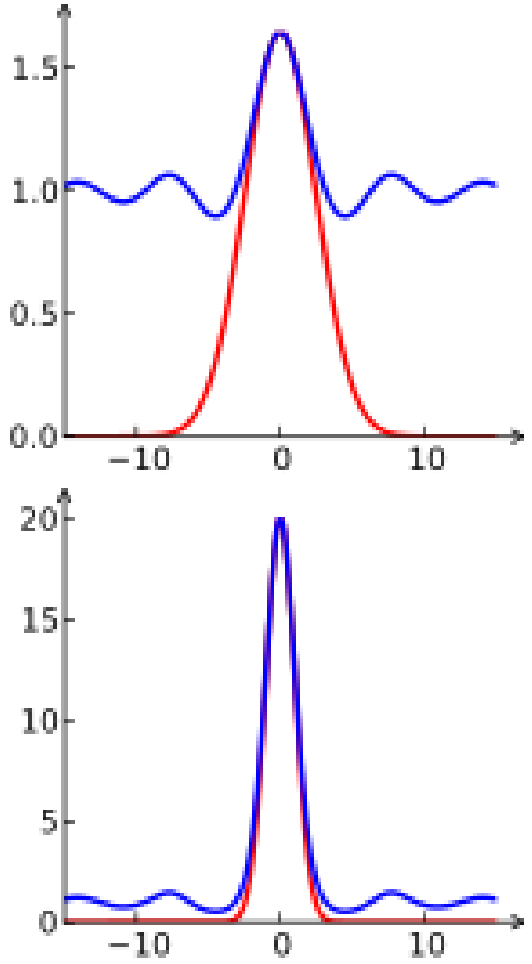


Image source: wikimedia

Importance weights follow a logarithmic distribution

This is a good news for us because having few important parameters means that we can use a strong regularization coefficient for them while having enough free capacity to learn new tasks.

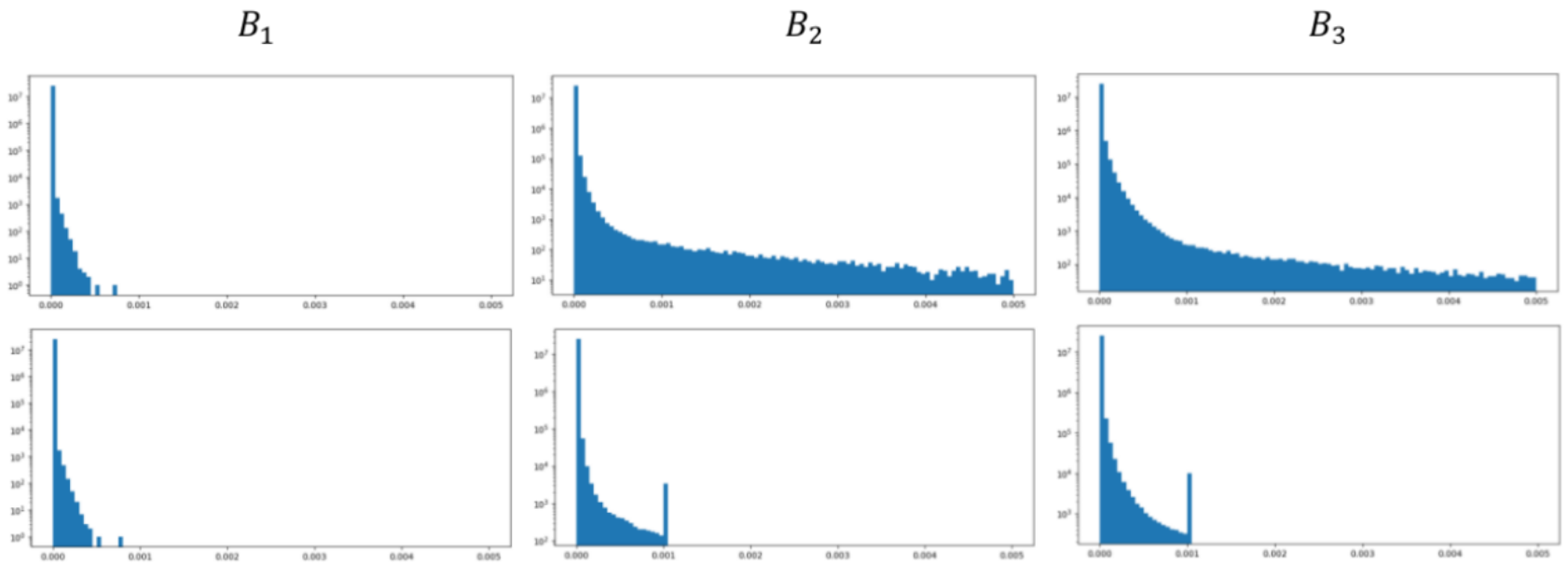


Figure 4: CaffeNet trained by EWC on CORe50 SIT (details in Section 4.2). The first row shows F values distribution denoting a long tail on the right: considering the logarithmic scale, the number of weights values taking high values in F is quite limited. The second row shows the normalized matrix $\hat{F}$ obtained by averaging F values and max clipping to 0.001. Saturation to 0.001 is well evident, but after B_3 the fraction of saturated weights is small (about 1/1000).

Continuous Learning in Single-Incremental Tasks, Maltoni & Lomonaco, Neural Networks, 2019.

Recap

- EWC provides a good proxy loss for past experiences
- Computing the Fisher requires only the gradients (computed at boundaries)

Problems:

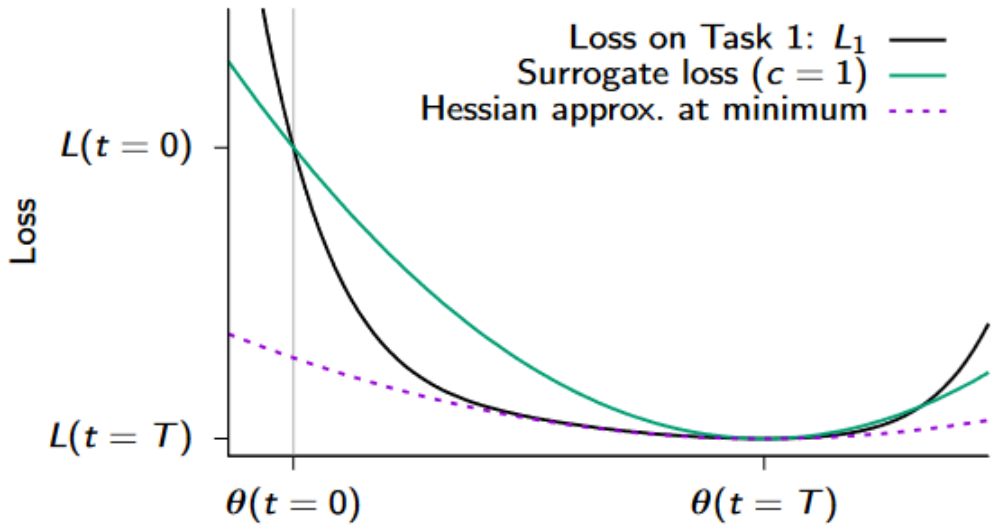
- Needs task boundaries and large batches, which makes it useless in online settings
- In «separate» mode it has a linear cost in the number of tasks (one Fisher and one model stored for each task)

Synaptic Intelligence

Online estimate of the curvature of the loss with Synaptic Intelligence

Online CL – Importance-based Methods

- EWC requires a trained model (i.e. after convergence) to estimate the Fisher Information.
  - We need task boundaries
  - We need enough data to guarantee convergence
- Only applicable to batch scenarios.
- We need an online importance estimate.



Synaptic Intelligence (SI)

KEY IDEA: a parameter importance is proportional to its contribution to the loss decrease over the entire training trajectory

- How do we measure it? The change in the loss is approximated using the gradient (for small steps)
- $\theta(t)$ weights at time t, $\delta(t)$ change in weights, $g_k(t)$ gradient for weight k

$$L(\boldsymbol{\theta}(t) + \boldsymbol{\delta}(t)) - L(\boldsymbol{\theta}(t)) \approx \sum_k g_k(t)\delta_k(t) \ , \qquad (1)$$

Continual Learning Through Synaptic Intelligence, Zenke et al, 2017.
Continuous Learning in Single-Incremental Tasks, Maltoni & Lomonaco, Neural Networks, 2019.

Synaptic Intelligence (SI)

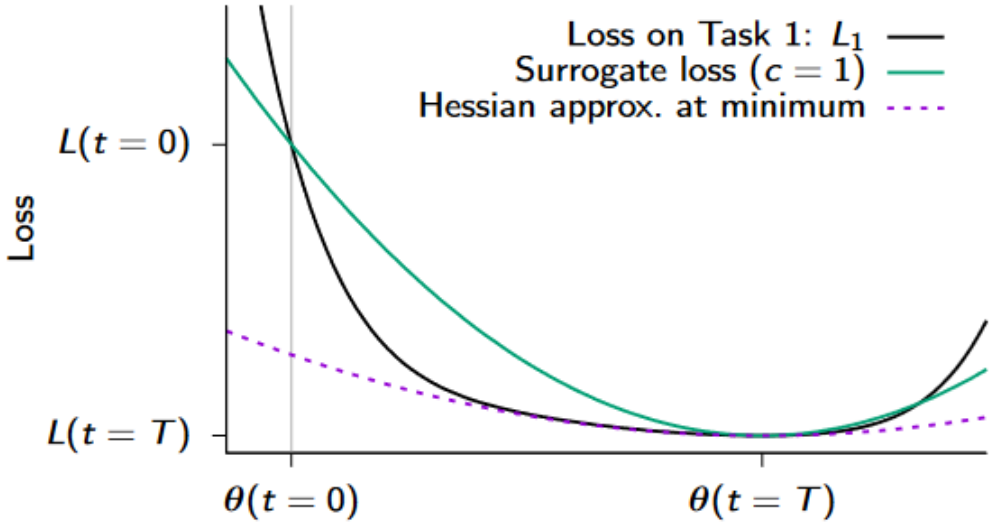
- The total contribution to the loss change is the sum of weight changes, weighted by the gradient, over the entire training curve (equation below)
- $\theta'(t)$ weight change
- ω_k^μ importance of weight k for task μ

$$\int_{t^{\mu-1}}^{t^\mu} \boldsymbol{g}(\boldsymbol{\theta}(t)) \cdot \boldsymbol{\theta}'(t) dt = \sum_k \int_{t^{\mu-1}}^{t^\mu} g_k(\theta(t)) \theta_k'(t) dt$$
$$\equiv -\sum_k \omega_k^\mu. \quad (3)$$

Continual Learning Through Synaptic Intelligence, Zenke et al, 2017.
Continuous Learning in Single-Incremental Tasks, Maltoni & Lomonaco, Neural Networks, 2019.

Synaptic Intelligence (SI)

- The loss penalize changes for important parameters
- Similar to EWC with different importance values
- c regularization strength
- $\Delta_k^{\nu} \equiv \theta_k(t^{\nu}) - \theta_k(t^{\nu-1})$ is the distance between the parameter k after exp. ν and $\nu - 1$
- ξ is a small number in the denominator that avoids numerical issues



$$\tilde{L}_{\mu} = L_{\mu} + \underbrace{c \sum_k \Omega_k^{\mu} \left(\tilde{\theta}_k - \theta_k\right)^2}_{\text{surrogate loss}} \qquad (4)$$

$$\Omega_k^{\mu} = \sum_{\nu<\mu} \frac{\omega_k^{\nu}}{(\Delta_k^{\nu})^2 + \xi} \,.$$

Continual Learning Through Synaptic Intelligence, Zenke et al, 2017.
Continuous Learning in Single-Incremental Tasks, Maltoni & Lomonaco, Neural Networks, 2019.

Synaptic Intelligence (SI)

KEY IDEA: a parameter importance is proportional to its contribution to the loss decrease over the entire training trajectory

- efficient online computation
- As with EWC, hyperparameters may be difficult to calibrate

$$L(\boldsymbol{\theta}(t) + \boldsymbol{\delta}(t)) - L(\boldsymbol{\theta}(t)) \approx \sum_k g_k(t)\delta_k(t)\,, \qquad (1)$$

$$\int_C \boldsymbol{g}(\boldsymbol{\theta}(t))d\boldsymbol{\theta} = \int_{t_0}^{t_1} \boldsymbol{g}(\boldsymbol{\theta}(t)) \cdot \boldsymbol{\theta}'(t)dt. \qquad (2)$$

$$\begin{aligned} \int_{t^{\mu-1}}^{t^\mu} \boldsymbol{g}(\boldsymbol{\theta}(t)) \cdot \boldsymbol{\theta}'(t)dt &= \sum_k \int_{t^{\mu-1}}^{t^\mu} g_k(\theta(t))\theta_k'(t)dt \\ &\equiv -\sum_k \omega_k^\mu. \end{aligned} \qquad (3)$$

$$\tilde{L}_\mu = L_\mu + \underbrace{c\sum_k \Omega_k^\mu \left(\tilde{\theta}_k - \theta_k\right)^2}_{\text{surrogate loss}} \qquad (4)$$

$$\Omega_k^\mu = \sum_{\nu<\mu} \frac{\omega_k^\nu}{(\Delta_k^\nu)^2 + \xi}\,. \qquad (5)$$

Continual Learning Through Synaptic Intelligence, Zenke et al, 2017.
Continuous Learning in Single-Incremental Tasks, Maltoni & Lomonaco, Neural Networks, 2019.

Take-Home Messages

- In replay-free CL, the key issue is the approximation of the loss for the old experiences without direct access to their data
- We can approximate it with a quadratic loss using the previous model and an estimate of the curvature
- **EWC**: distance in the “information geometry” manifold
- **Synaptic Intelligence**: online approximation of the curvature of the loss function
- Prior-based methods are quite effective with
  - Small drifts (like some domain-incremental scenarios)
  - Pretrained models
- In many scenarios (e.g. class-incremental), the assumptions of prior-based methods (flat minima, linear approximations) are too strong and the method fail to beat naïve finetuning

References

- «Limitations of the Empirical Fisher Approximation for Natural Gradient Descent» Frederik Kunstner, Lukas Balles, Philipp Hennig https://arxiv.org/abs/1905.12558
- A nice blogpost: https://andrewcharlesjones.github.io/journal/natural-gradients.html
- a stable implementation of Fisher computations (diagonal, full and factorized): https://nngeometry.readthedocs.io/en/latest/

Next Lecture

More regularization methods

- CL and SGD bias
- Sparsity
- Knowledge distillation
- CL as constraints