



Imports

| root package |
|---|
| dataset representation and loading |
| |
| |
| computation graph |
| variable node in computation graph |
| neural networks |
| layers, activations and more |
| optimizers e.g. gradient desc, ADAM, etc |
| |
| vision datasets, architectures & transforms |
| composable transforms |
| |
| distributed comunication |
| memory sharing processes |
| |

Tensors

| lensors | | |
|------------------------------|---|--|
| Creation | | |
| torch.randn(*size) | tensor with independent N(0,1) entries | |
| torch.[ones zeros](*size) | tensor with all 1's [or 0's] | |
| torch.Tensor(L) | create tensor from [nested] list or ndarray L | |
| x.clone() | clone of x | |
| Dimensionality | | |
| x.size() | return tuple-like object of dimensions | |
| torch.cat(tensor_seq, dim=0) | concatenates tensors along dim | |
| x.view(a,b,) | reshapes x into size (a,b,) | |
| x.view(-1,a) | reshapes x into size (b,a) for some b | |
| x.transpose(a,b) | swaps dimensions a and b | |
| x.permute(*dims) | permutes dimensions | |
| x.unsqueeze(dim) | tensor with added axis | |
| x.unsqueeze(dim=2) | (a,b,c) tensor -> (a,b,1,c) tensor | |
| Algebra | | |
| A.mm(B) | matrix multiplication | |
| A.mv(x) | matrix-vector multiplication | |
| x.t() | matrix transpose | |
| GPU | | |
| torch.cuda.is_available() | check for cuda | |
| x.cuda() | move x's data from CPU to GPU and return new object | |
| x.cpu() | move x's data from GPU to CPU and return new object | |

Deep Learning

| Layers | |
|---|---|
| nn.Linear(m,n) | fully connected layer from m to n units |
| nn.ConvXd(m, n, s) | X dimensional conv layer from m to n channels where $X \in \{1,2,3\}$ and kernel size is s |
| nn.MaxPoolXd(s) | X dimensional pooling layer (notation as above) |
| nn.BatchNorm | batch norm layer |
| nn.RNN/LSTM/GRU | recurrent layers |
| nn.Dropout(p=0.5, inplace=False) | dropout layer for any dimensional input |
| nn.Dropout2d(p=0.5, inplace=False) | 2-dimensional channel-wise dropout |
| nn.Embedding(num_embeddings, embedding_dim) | (tensor-wise) mapping from indices to embedding vectors |
| Loss functions | |
| nn.X where for example X is | BCELoss, CrossEntropyLoss, L1Loss, MSELoss, NLLLoss SoftMarginLoss, MultiLabelSoftMarginLoss, CosineEm |
| | |
| Activation functions | beddingLoss, KLDivLoss, MarginRankingLoss, HingeEm |
| Activation functions nn.X where for example X is | beddingLoss, KLDivLoss, MarginRankingLoss, HingeEmbeddingLoss or CosineEmbeddingLoss ReLU, ReLU6, ELU, SELU, PReLU, LeakyReLU, Threshold, Hardtanh, Sigmoid, Tanh, LogSigmoid, Softplus, Soft |
| nn.X where for example X is | beddingLoss, KLDivLoss, MarginRankingLoss, HingeEmbeddingLoss or CosineEmbeddingLoss ReLU, ReLU6, ELU, SELU, PReLU, LeakyReLU, Threshold, Hardtanh, Sigmoid, Tanh, LogSigmoid, Softplus, Softshrink, Softsign, TanhShrink, Softmin, Softmax, Soft |
| nn.X where for example X is Optimizers | beddingLoss, KLDivLoss, MarginRankingLoss, HingeEmbeddingLoss or CosineEmbeddingLoss ReLU, ReLU6, ELU, SELU, PReLU, LeakyReLU, Threshold, Hardtanh, Sigmoid, Tanh, LogSigmoid, Softplus, Softshrink, Softsign, TanhShrink, Softmin, Softmax, Softmax2d or LogSoftmax |
| Optimizers opt = optim.X(model.parameters(),) | beddingLoss, KLDivLoss, MarginRankingLoss, HingeEmbeddingLoss or CosineEmbeddingLoss ReLU, ReLU6, ELU, SELU, PReLU, LeakyReLU, Threshold, Hardtanh, Sigmoid, Tanh, LogSigmoid, Softplus, Softshrink, Softsign, TanhShrink, Softmin, Softmax, Softmax2d or LogSoftmax create optimizer |
| nn.X where for example X is Optimizers | beddingLoss, KLDivLoss, MarginRankingLoss, HingeEmbeddingLoss or CosineEmbeddingLoss ReLU, ReLU6, ELU, SELU, PReLU, LeakyReLU, Threshold, Hardtanh, Sigmoid, Tanh, LogSigmoid, Softplus, Softshrink, Softsign, TanhShrink, Softmin, Softmax, Softmax2d or LogSoftmax create optimizer update weights |
| Optimizers opt = optim.X(model.parameters(),) optistep() optim.X where for example X is | beddingLoss, KLDivLoss, MarginRankingLoss, HingeEmbeddingLoss or CosineEmbeddingLoss ReLU, ReLU6, ELU, SELU, PReLU, LeakyReLU, Threshold, Hardtanh, Sigmoid, Tanh, LogSigmoid, Softplus, Softshrink, Softsign, TanhShrink, Softmin, Softmax, Softmax2d or LogSoftmax create optimizer update weights SGD, Adadelta, Adagrad, Adam, SparseAdam, Adamax |
| Optimizers opt = optim.X(model.parameters(),) optistep() optim.X where for example X is Learning rate scheduling | beddingLoss, KLDivLoss, MarginRankingLoss, HingeEmbeddingLoss or CosineEmbeddingLoss ReLU, ReLU6, ELU, SELU, PReLU, LeakyReLU, Threshold, Hardtanh, Sigmoid, Tanh, LogSigmoid, Softplus, Softshrink, Softsign, TanhShrink, Softmin, Softmax, Softmax2d or LogSoftmax create optimizer update weights SGD, Adadelta, Adagrad, Adam, SparseAdam, Adamax ASGD, LBFGS, RMSProp or Rprop |
| Optimizers opt = optim.X(model.parameters(),) optistep() optim.X where for example X is | beddingLoss, KLDivLoss, MarginRankingLoss, HingeEmbeddingLoss or CosineEmbeddingLoss ReLU, ReLU6, ELU, SELU, PReLU, LeakyReLU, Threshold, Hardtanh, Sigmoid, Tanh, LogSigmoid, Softplus, Softshrink, Softsign, TanhShrink, Softmin, Softmax, Softmax2d or LogSoftmax create optimizer update weights SGD, Adadelta, Adagrad, Adam, SparseAdam, Adamax |

Data - torch.utils.data.X

| Datasets | |
|--|--|
| Dataset | abstract class representing data set |
| TensorDataset | labelled data set in the form of tensors |
| ConcatDataset | concatation of iterable of Datasets |
| DataLoaders and DataSamplers DataLoader(dataset, batch_size=1,) | loads data batches agnostically of structure of individual data points |
| sampler.Sampler(dataset,) | abstract class dealing with ways to sample from dataset |
| sampler.XSampler where | Sequential, Random, Subset, WeightedRandom or Distributed |