Python numpy 模块，sqrt() 实例源码

我们从Python开源项目中，提取了以下**50**个代码示例，用于说明如何使用**numpy.sqrt()**。

项目： GELUs **作者：** hendrycks | 项目源码 (https://github.com/hendrycks/GELUs) | 文件源码 (https://github.com/hendrycks/GELUs/tree/master/nn.py)

```
def gelu(x):
    return 0.5 * x * (1 + T.tanh(T.sqrt(2 / np.pi) * (x + 0.044715 * T.pow(x, 3))))
```


项目： qqmbr **作者：** ischurov | 项目源码

(https://github.com/ischurov/qqmbr) | 文件源码
(https://github.com/ischurov/qqmbr/tree/master/qqmbr/odebook.py)

```
def normvectorfield(xs,ys,fs,**kw):
    """
    plot normalized vector field

    kwargs
    ======

    - length is a desired length of the lines (default: 1)
    - the rest of kwards are passed to plot
    """
    length = kw.pop('length') if 'length' in kw else 1
    x, y = np.meshgrid(xs, ys)
    # calculate vector field
    vx,vy = fs(x,y)
    # plot vecor field
    norm = length /np.sqrt(vx**2+vy**2)
    plt.quiver(x, y, vx * norm, vy * norm, angles='xy',**kw)
```

项目：GELUs　**作者：**hendrycks　| 项目源码 (https://github.com/hendrycks/GELUs) | 文件源码 (https://github.com/hendrycks/GELUs/tree/master/nn.py)

```
def adam_updates(params, cost, lr=0.001, mom1=0.9, mom2=0.999):
    updates = []
    grads = T.grad(cost, params)
    t = th.shared(np.cast[th.config.floatX](1.))
    for p, g in zip(params, grads):
        v = th.shared(np.cast[th.config.floatX](p.get_value() * 0.))
        mg = th.shared(np.cast[th.config.floatX](p.get_value() * 0.))
        v_t = mom1*v + (1. - mom1)*g
        mg_t = mom2*mg + (1. - mom2)*T.square(g)
        v_hat = v_t / (1. - mom1 ** t)
        mg_hat = mg_t / (1. - mom2 ** t)
        g_t = v_hat / T.sqrt(mg_hat + 1e-8)
        p_t = p - lr * g_t
        updates.append((v, v_t))
        updates.append((mg, mg_t))
        updates.append((p, p_t))
    updates.append((t, t+1))
    return updates
```

项目：cnn-graph-classification　**作者：**giannisnik　| 项目源码 (https://github.com/giannisnik/cnn-graph-classification) | 文件源码 (https://github.com/giannisnik/cnn-graph-classification/tree/master/kcnn/nystrom.py)

```
def fit(self, graphs, y=None):
        rnd = check_random_state(self.random_state)
        n_samples = len(graphs)

        # get basis vectors
        if self.n_components > n_samples:
            n_components = n_samples
        else:
            n_components = self.n_components
        n_components = min(n_samples, n_components)
        inds = rnd.permutation(n_samples)
        basis_inds = inds[:n_components]
        basis = []
```



```
                                                **self._get_kernel_params(

        # sqrt of kernel matrix on basis vectors
        U, S, V = svd(basis_kernel)
        S = np.maximum(S, 1e-12)
        self.normalization_ = np.dot(U * 1. / np.sqrt(S), V)
        self.components_ = basis
        self.component_indices_ = inds
        return self
```

项目： pylspm　**作者：** lseman　| 项目源码 (https://github.com/lseman/pylspm) | 文件源码 (https://github.com/lseman/pylspm/tree/master/pylspm/pylspm.py)

```
def alpha(self):
        # Cronbach Alpha
        alpha = pd.DataFrame(0, index=np.arange(1), columns=self.latent)

        for i in range(self.lenlatent):
            block = self.data_[self.Variables['measurement']
                               [self.Variables['latent'] == self.latent[i]]
]
            p = len(block.columns)

            if(p != 1):
                p_ = len(block)
                correction = np.sqrt((p_ - 1) / p_)
                soma = np.var(np.sum(block, axis=1))
                cor_ = pd.DataFrame.corr(block)

                denominador = soma * correction**2
                numerador = 2 * np.sum(np.tril(cor_) - np.diag(np.diag(cor_
)))

                alpha_ = (numerador / denominador) * (p / (p - 1))
                alpha[self.latent[i]] = alpha_
            else:
                alpha[self.latent[i]] = 1

        return alpha.T
```

项目： GYM_DRL　**作者：** Kyushik　| 项目源码 (https://github.com/Kyushik/GYM_DRL) | 文件源码 (https://github.com/Kyushik/GYM_DRL/tree/master/CartPole_DRQN.py)

```
def xavier_initializer(shape):
    dim_sum = np.sum(shape)
    if len(shape) == 1:
        dim_sum += 1
    bound = np.sqrt(2.0 / dim_sum)
    return tf.random_uniform(shape, minval=-bound, maxval=bound)

# # Assigning network variables to target network variables
# def assign_network_to_target():
#   update_wfc = tf.assign(w_fc_target, w_fc)
#   update_bfc = tf.assign(b_fc_target, b_fc)

#   sess.run(update_wfc)
#   sess.run(update_bfc)

#   cell_target = cell

# Input
```


项目：uwb_tracker_ros　**作者：**eth-ait　| 项目源码 (https://github.com/eth-ait/uwb_tracker_ros) | 文件源码 (https://github.com/eth-ait/uwb_tracker_ros/tree/master/src/uwb_tracker_node.py)

```
def _ikf_iteration(self, x, n, ranges, h, H, z, estimate, R):
        """Update tracker based on a multi-range message.

        Args:
             multi_range_msg (uwb.msg.UWBMultiRangeWithOffsets): ROS multi-ra



        Returns:
            new_estimate (StateEstimate): Updated position estimate.
        """
        new_position = n[0:3]
        self._compute_measurements_and_jacobians(ranges, new_position, h, H
, z)
        res = z - h
        S = np.dot(np.dot(H, estimate.covariance), H.T) + R
        K = np.dot(estimate.covariance, self._solve_equation_least_squares(
S.T, H).T)
        mahalanobis = np.sqrt(np.dot(self._solve_equation_least_squares(S.T
, res).T, res))
        if res.size not in self.outlier_thresholds:
            self.outlier_thresholds[res.size] = scipy.stats.chi2.isf(self.o
utlier_threshold_quantile, res.size)
        outlier_threshold = self.outlier_thresholds[res.size]
        if mahalanobis < outlier_threshold:
            n = x + np.dot(K, (res - np.dot(H, x - n)))
            outlier_flag = False
        else:
            outlier_flag = True
        return n, K, outlier_flag
```

项目：YellowFin_Pytorch　**作者：**JianGoForIt　| 项目源码 (https://github.com/JianGoForIt/YellowFin_Pytorch) | 文件源码 (https://github.com/JianGoForIt/YellowFin_Pytorch/tree/master/tuner_utils/yellowfin.py)

```
def dist_to_opt(self):
    global_state = self._global_state
    beta = self._beta
    if self._iter == 0:
      global_state["grad_norm_avg"] = 0.0
      global_state["dist_to_opt_avg"] = 0.0
    global_state["grad_norm_avg"] = \
      global_state["grad_norm_avg"] * beta + (1 - beta) * math.sqrt(global_
state["grad_norm_squared"] )
    global_state["dist_to_opt_avg"] = \
      global_state["dist_to_opt_avg"] * beta \
      + (1 - beta) * global_state["grad_norm_avg"] / (global_state['grad_no
rm_squared_avg'] + eps)
    if self._zero_debias:
      debias_factor = self.zero_debias_factor()
      self._dist_to_opt = global_state["dist_to_opt_avg"] / debias_factor
    else:
      self._dist_to_opt = global_state["dist_to_opt_avg"]
    if self._sparsity_debias:
      self._dist_to_opt /= (np.sqrt(self._sparsity_avg) + eps)
    return
```


项目：YellowFin_Pytorch　作者：JianGoForIt　| 项目源码

(https://github.com/JianGoForIt/YellowFin_Pytorch) | 文件源码 (https://github.com/JianGoForIt/YellowFin_Pytorch/tree/master/tuner_utils/yellowfin.py)

```
def lr_grad_norm_avg(self):
    # this is for enforcing lr * grad_norm not
    # increasing dramatically in case of instability.
    #  Not necessary for basic use.
    global_state = self._global_state
    beta = self._beta
    if "lr_grad_norm_avg" not in global_state:
      global_state['grad_norm_squared_avg_log'] = 0.0
    global_state['grad_norm_squared_avg_log'] = \
      global_state['grad_norm_squared_avg_log'] * beta \
      + (1 - beta) * np.log(global_state['grad_norm_squared'] + eps)
    if "lr_grad_norm_avg" not in global_state:
      global_state["lr_grad_norm_avg"] = \
        0.0 * beta + (1 - beta) * np.log(self._lr * np.sqrt(global_state['g
rad_norm_squared'] ) + eps)
      # we monitor the minimal smoothed ||lr * grad||
      global_state["lr_grad_norm_avg_min"] = \
        np.exp(global_state["lr_grad_norm_avg"] / self.zero_debias_factor()
 )
    else:
      global_state["lr_grad_norm_avg"] = global_state["lr_grad_norm_avg"] *
 beta \
        + (1 - beta) * np.log(self._lr * np.sqrt(global_state['grad_norm_sq
uared'] ) + eps)
      global_state["lr_grad_norm_avg_min"] = \
        min(global_state["lr_grad_norm_avg_min"],
            np.exp(global_state["lr_grad_norm_avg"] / self.zero_debias_fact
or() ) )
```

项目：YellowFin_Pytorch　**作者：**JianGoForIt　| 项目源码 (https://github.com/JianGoForIt/YellowFin_Pytorch) | 文件源码 (https://github.com/JianGoForIt/YellowFin_Pytorch/tree/master/numerical_test/YFDebug/yellowfin_backup.py)

```
def dist_to_opt(self):
    global_state = self._global_state
    beta = self._beta
    if self._iter == 0:
      global_state["grad_norm_avg"] = 0.0
      global_state["dist_to_opt_avg"] = 0.0
    global_state["grad_norm_avg"] = \
      global_state["grad_norm_avg"] * beta + (1 - beta) * math.sqrt(global_
state["grad_norm_squared"] )
    global_state["dist_to_opt_avg"] = \
      global_state["dist_to_opt_avg"] * beta \
      + (1 - beta) * global_state["grad_norm_avg"] / (global_state['grad_no
rm_squared_avg'] + eps)
    if self._zero_debias:
      debias_factor = self.zero_debias_factor()
      self._dist_to_opt = global_state["dist_to_opt_avg"] / debias_factor
    else:
      self._dist_to_opt = global_state["dist_to_opt_avg"]
    if self._sparsity_debias:
      self._dist_to_opt /= (np.sqrt(self._sparsity_avg) + eps)
    return
```

项目：YellowFin_Pytorch　**作者：**JianGoForIt　| 项目源码 (https://github.com/JianGoForIt/YellowFin_Pytorch) | 文件源码 (https://github.com/JianGoForIt/YellowFin_Pytorch/tree/master/numerical_test/YFDebug/yellowfin_backup.py)

```
def lr_grad_norm_avg(self):
    # this is for enforcing lr * grad_norm not
    # increasing dramatically in case of instability.
    #  Not necessary for basic use.
    global_state = self._global_state
    beta = self._beta
    if "lr_grad_norm_avg" not in global_state:
      global_state['grad_norm_squared_avg_log'] = 0.0
    global_state['grad_norm_squared_avg_log'] = \
      global_state['grad_norm_squared_avg_log'] * beta \
      + (1 - beta) * np.log(global_state['grad_norm_squared'] + eps)
    if "lr_grad_norm_avg" not in global_state:
      global_state["lr_grad_norm_avg"] = \
        0.0 * beta + (1 - beta) * np.log(self._lr * np.sqrt(global_state['g
rad_norm_squared'] ) + eps)
      # we monitor the minimal smoothed ||lr * grad||
      global_state["lr_grad_norm_avg_min"] = \
        np.exp(global_state["lr_grad_norm_avg"] / self.zero_debias_factor()
 )
    else:
      global_state["lr_grad_norm_avg"] = global_state["lr_grad_norm_avg"] *
 beta \
        + (1 - beta) * np.log(self._lr * np.sqrt(global_state['grad_norm_sq
uared'] ) + eps)
      global_state["lr_grad_norm_avg_min"] = \
        min(global_state["lr_grad_norm_avg_min"],
            np.exp(global_state["lr_grad_norm_avg"] / self.zero_debias_fact
or() ) )
```

项目： YellowFin_Pytorch　**作者：** JianGoForIt　| 项目源码 (https://github.com/JianGoForIt/YellowFin_Pytorch) | 文件源码 (https://github.com/JianGoForIt/YellowFin_Pytorch/tree/master/numerical_test/YFDebug/yellowfin_backup.py)

```
def get_cubic_root(self):
    # We have the equation x^2 D^2 + (1-x)^4 * C / h_min^2
    # where x = sqrt(mu).
    # We substitute x, which is sqrt(mu), with x = y + 1.
    # It gives y^3 + py = q
    # where p = (D^2 h_min^2)/(2*C) and q = -p.
    # We use the Vieta's substution to compute the root.
    # There is only one real solution y (which is in [0, 1] ).
    # http://mathworld.wolfram.com/VietasSubstitution.html
    # eps in the numerator is to prevent momentum = 1 in case of zero gradi
ent
    p = (self._dist_to_opt + eps)**2 * (self._h_min + eps)**2 / 2 / (self._
grad_var + eps)
    w3 = (-math.sqrt(p**2 + 4.0 / 27.0 * p**3) - p) / 2.0
    w = math.copysign(1.0, w3) * math.pow(math.fabs(w3), 1.0/3.0)
    y = w - p / 3.0 / (w + eps)
    x = y + 1

    if DEBUG:
      logging.debug("p %f, den %f", p, self._grad_var + eps)
      logging.debug("w3 %f ", w3)
      logging.debug("y %f, den %f", y, w + eps)

    return x
```

项目：YellowFin_Pytorch　**作者：**JianGoForIt　| 项目源码 (https://github.com/JianGoForIt/YellowFin_Pytorch) | 文件源码 (https://github.com/JianGoForIt/YellowFin_Pytorch/tree/master/numerical_test/YFDebug/yellowfin_backup.py)

```
def update_hyper_param(self):
    for group in self._optimizer.param_groups:
      group['momentum'] = self._mu
      if self._force_non_inc_step == False:
        group['lr'] = min(self._lr * self._lr_factor,
          self._lr_grad_norm_thresh / (math.sqrt(self._global_state["grad_n
orm_squared"] ) + eps) )
      elif self._iter > self._curv_win_width:
        # force to guarantee lr * grad_norm not increasing dramatically.
        # Not necessary for basic use. Please refer to the comments
        # in YFOptimizer.__init__ for more details
        self.lr_grad_norm_avg()
        debias_factor = self.zero_debias_factor()
        group['lr'] = min(self._lr * self._lr_factor,
          2.0 * self._global_state["lr_grad_norm_avg_min"] \
          / (np.sqrt(np.exp(self._global_state['grad_norm_squared_avg_log']
 / debias_factor) ) + eps) )
    return
```

项目： treecat　**作者：** posterior　| 项目源码 (https://github.com/posterior/treecat) | 文件源码 (https://github.com/posterior/treecat/tree/master/treecat/serving.py)

```
def latent_correlation(self):
        """Compute correlation matrix among latent features.

        This computes the generalization of Pearson's correlation to discrete

        data. Let I(X;Y) be the mutual information. Then define correlation a


          rho(X,Y) = sqrt(1 - exp(-2 I(X;Y)))

        Returns:
            A [V, V]-shaped numpy array of feature-feature correlations.
        """
        result = self._ensemble[0].latent_correlation()
        for server in self._ensemble[1:]:
            result += server.latent_correlation()
        result /= len(self._ensemble)
        return result
```


项目：spyking-circus　**作者：**spyking-circus　| 项目源码 (https://github.com/spyking-circus/spyking-circus) | 文件源码 (https://github.com/spyking-circus/spyking-

circus/tree/master/circus/shared/utils.py)

```
def fit_transform(self, X, y=None):
        """Fit the model with X and apply the dimensionality reduction on X
.
        Parameters
        ----------
        X : array-like, shape (n_samples, n_features)
            Training data, where n_samples is the number of samples
            and n_features is the number of features.
        Returns
        -------
        X_new : array-like, shape (n_samples, n_components)
        """
        U, S, V = self._fit(X)
        U = U[:, :int(self.n_components_)]

        if self.whiten:
            # X_new = X * V / S * sqrt(n_samples) = U * sqrt(n_samples)
            U *= sqrt(X.shape[0])
        else:
            # X_new = X * V = U * S * V^T * V = U * S
            U *= S[:int(self.n_components_)]

        return U
```

项目：spyking-circus　**作者：**spyking-circus　| 项目源码 (https://github.com/spyking-circus/spyking-circus) | 文件源码 (https://github.com/spyking-circus/spyking-circus/tree/master/circus/shared/utils.py)

```
def get_covariance(self):
        """Compute data covariance with the generative model.
        ``cov = components_.T * S**2 * components_ + sigma2 * eye(n_features)

        where  S**2 contains the explained variances.
        Returns
        -------
        cov : array, shape=(n_features, n_features)
            Estimated covariance of data.
        """
        components_ = self.components_
        exp_var = self.explained_variance_
        if self.whiten:
            components_ = components_ * np.sqrt(exp_var[:, np.newaxis])
        exp_var_diff = np.maximum(exp_var - self.noise_variance_, 0.)
        cov = np.dot(components_.T * exp_var_diff, components_)
        cov.flat[::len(cov) + 1] += self.noise_variance_  # modify diag inp
lace
        return cov
```

项目：spyking-circus　**作者：**spyking-circus　| 项目源码 (https://github.com/spyking-circus/spyking-circus) | 文件源码 (https://github.com/spyking-circus/spyking-circus/tree/master/circus/shared/utils.py)

```
def inverse_transform(self, X):
        """Transform data back to its original space, i.e.,
        return an input X_original whose transform would be X
        Parameters
        ----------
        X : array-like, shape (n_samples, n_components)
            New data, where n_samples is the number of samples
            and n_components is the number of components.
        Returns
        -------
        X_original array-like, shape (n_samples, n_features)
        """
        check_is_fitted(self, 'mean_')

        if self.whiten:
            return fast_dot(
                X,
                np.sqrt(self.explained_variance_[:, np.newaxis]) *
                self.components_) + self.mean_
        else:
            return fast_dot(X, self.components_) + self.mean_
```

项目： MKLMM　**作者：** omerwe　| 项目源码 (https://github.com/omerwe/MKLMM) | 文件源码 (https://github.com/omerwe/MKLMM/tree/master/mklmm.py)

```
def infExact_scipy_post(self, K, covars, y, sig2e, fixedEffects):
        n = y.shape[0]

        #mean vector
        m = covars.dot(fixedEffects)

        if (K.shape[1] < K.shape[0]): K_true = K.dot(K.T)
        else: K_true = K

        if sig2e<1e-6:
            L = la.cholesky(K_true + sig2e*np.eye(n), overwrite_a=True, che
ck_finite=False)      #Cholesky factor of covariance with noise
            sl =   1
            pL = -self.solveChol(L, np.eye(n))
                      #L = -inv(K+inv(sW^2))
        else:
            L = la.cholesky(K_true/sig2e + np.eye(n), overwrite_a=True, che
ck_finite=False)      #Cholesky factor of B
            sl = sig2e
            pL = L
                      #L = chol(eye(n)+sW*sW'.*K)
        alpha = self.solveChol(L, y-m, overwrite_b=False) / sl

        post = dict([])
        post['alpha'] = alpha
                     #return the posterior parameters
        post['sW'] = np.ones(n) / np.sqrt(sig2e)
                     #sqrt of noise precision vector
        post['L'] = pL
        return post
```

项目： MKLMM　**作者：** omerwe　| 项目源码 (https://github.com/omerwe/MKLMM) | 文件源码 (https://github.com/omerwe/MKLMM/tree/master/gpUtils.py)

```
def removeTopPCs(X, numRemovePCs):
    t0 = time.time()
    X_mean = X.mean(axis=0)
    X -= X_mean
    XXT = symmetrize(blas.dsyrk(1.0, X, lower=0))
    s,U = la.eigh(XXT)
    if (np.min(s) < -1e-4): raise Exception('Negative eigenvalues found')
    s[s<0]=0
    ind = np.argsort(s)[::-1]
    U = U[:, ind]
    s = s[ind]
    s = np.sqrt(s)

    #remove null PCs
    ind = (s>1e-6)
    U = U[:, ind]
    s = s[ind]

    V = X.T.dot(U/s)
    #print 'max diff:', np.max(((U*s).dot(V.T) - X)**2)
    X = (U[:, numRemovePCs:]*s[numRemovePCs:]).dot((V.T)[numRemovePCs:, :])
    X += X_mean

    return X
```

(https://github.com/omerwe/MKLMM) | 文件源码
(https://github.com/omerwe/MKLMM/tree/master/gpUtils.py)

```
def normalizeSNPs(normMethod, X, y, prev=None, frqFile=None):
    if (normMethod == 'frq'):
        print 'flipping SNPs for standardization...'
        empMean = X.mean(axis=0) / 2.0
        X[:, empMean>0.5] = 2 - X[:, empMean>0.5]
        mafs = np.loadtxt(frqFile, usecols=[1,2]).mean(axis=1)
        snpsMean = 2*mafs
        snpsStd = np.sqrt(2*mafs*(1-mafs))
    elif (normMethod == 'controls'):
        controls = (y<y.mean())
        cases = ~controls
        snpsMeanControls, snpsStdControls = X[controls, :].mean(axis=0), X[
controls, :].std(axis=0)
        snpsMeanCases, snpsStdCases = X[cases, :].mean(axis=0), X[cases, :]
.std(axis=0)
        snpsMean = (1-prev)*snpsMeanControls + prev*snpsMeanCases
        snpsStd = (1-prev)*snpsStdControls + prev*snpsStdCases
    elif (normMethod is None): snpsMean, snpsStd = X.mean(axis=0), X.std(ax
is=0)
    else: raise Exception('Unrecognized normalization method: ' + normMetho
d)

    return snpsMean, snpsStd
```

项目：human-rl　**作者**：gsastry　| 项目源码
(https://github.com/gsastry/human-rl) | 文件源码
(https://github.com/gsastry/human-rl/tree/master/scripts/universe_starter_agent/model.py)

```
def conv2d(x, num_filters, name, filter_size=(3, 3), stride=(1, 1), pad="SA
ME", dtype=tf.float32, collections=None):
    with tf.variable_scope(name):
        stride_shape = [1, stride[0], stride[1], 1]
        filter_shape = [filter_size[0], filter_size[1], int(x.get_shape()[3
]), num_filters]

        # there are "num input feature maps * filter height * filter width"
        # inputs to each hidden unit
        fan_in = np.prod(filter_shape[:3])
        # each unit in the lower layer receives a gradient from:
        # "num output feature maps * filter height * filter width" /
        #   pooling size
        fan_out = np.prod(filter_shape[:2]) * num_filters
        # initialize weights with random weights
        w_bound = np.sqrt(6. / (fan_in + fan_out))

        w = tf.get_variable("W", filter_shape, dtype, tf.random_uniform_ini
tializer(-w_bound, w_bound),
                            collections=collections)
        b = tf.get_variable("b", [1, 1, 1, num_filters], initializer=tf.con
stant_initializer(0.0),
                            collections=collections)
        return tf.nn.conv2d(x, w, stride_shape, pad) + b
```

项目：human-rl　**作者：**gsastry　| 项目源码 (https://github.com/gsastry/human-rl) | 文件源码 (https://github.com/gsastry/human-rl/tree/master/universe-starter-agent/model.py)

```
def conv2d(x, num_filters, name, filter_size=(3, 3), stride=(1, 1), pad="SA
ME", dtype=tf.float32, collections=None):
    with tf.variable_scope(name):
        stride_shape = [1, stride[0], stride[1], 1]
        filter_shape = [filter_size[0], filter_size[1], int(x.get_shape()[3
]), num_filters]

        # there are "num input feature maps * filter height * filter width"
        # inputs to each hidden unit
        fan_in = np.prod(filter_shape[:3])
        # each unit in the lower layer receives a gradient from:
        # "num output feature maps * filter height * filter width" /
        #   pooling size
        fan_out = np.prod(filter_shape[:2]) * num_filters
        # initialize weights with random weights
        w_bound = np.sqrt(6. / (fan_in + fan_out))

        w = tf.get_variable("W", filter_shape, dtype, tf.random_uniform_ini
tializer(-w_bound, w_bound),
                            collections=collections)
        b = tf.get_variable("b", [1, 1, 1, num_filters], initializer=tf.con
stant_initializer(0.0),
                            collections=collections)
        return tf.nn.conv2d(x, w, stride_shape, pad) + b
```

项目：distributional_perspective_on_RL　**作者：**Kiwoo　| 项目源码 (https://github.com/Kiwoo/distributional_perspective_on_RL) | 文件源码 (https://github.com/Kiwoo/distributional_perspective_on_RL/tree/master/tf_util.py)

```
def batchnorm(x, name, phase, updates, gamma=0.96):
    k = x.get_shape()[1]
    runningmean = tf.get_variable(name+"/mean", shape=[1, k], initializer=t
f.constant_initializer(0.0), trainable=False)
    runningvar = tf.get_variable(name+"/var", shape=[1, k], initializer=tf.
constant_initializer(1e-4), trainable=False)
    testy = (x - runningmean) / tf.sqrt(runningvar)

    mean_ = mean(x, axis=0, keepdims=True)
    var_ = mean(tf.square(x), axis=0, keepdims=True)
    std = tf.sqrt(var_)
    trainy = (x - mean_) / std

    updates.extend([
        tf.assign(runningmean, runningmean * gamma + mean_ * (1 - gamma)),
        tf.assign(runningvar, runningvar * gamma + var_ * (1 - gamma))
    ])

    y = switch(phase, trainy, testy)

    out = y * tf.get_variable(name+"/scaling", shape=[1, k], initializer=tf
.constant_initializer(1.0), trainable=True)\
            + tf.get_variable(name+"/translation", shape=[1,k], initializer
=tf.constant_initializer(0.0), trainable=True)
    return out

# ============================================================
# Mathematical utils
# ============================================================
```

项目：cellranger　**作者：**10XGenomics　| 项目源码 (https://github.com/10XGenomics/cellranger) | 文件源码 (https://github.com/10XGenomics/cellranger/tree/master/lib/python/cellranger/analysis/pca.py)

```
def normalize_and_transpose(matrix):
    matrix.tocsc()

    m = normalize_by_umi(matrix)

    # Use log counts
    m.data = np.log2(1 + m.data)

    # Transpose
    m = m.T

    # compute centering (mean) and scaling (stdev)
    (c,v) = summarize_columns(m)
    s = np.sqrt(v)

    return (m, c, s)
```

项目：cellranger　**作者：**10XGenomics　| 项目源码 (https://github.com/10XGenomics/cellranger) | 文件源码 (https://github.com/10XGenomics/cellranger/tree/master/lib/python/cellranger/stats.py)

```
def merge_filtered_metrics(filtered_metrics):
    result = {
        'filtered_bcs': 0,
        'filtered_bcs_lb': 0,
        'filtered_bcs_ub': 0,
        'max_filtered_bcs': 0,
        'filtered_bcs_var': 0,
        'filtered_bcs_cv': 0,
    }
    for i, fm in enumerate(filtered_metrics):
        # Add per-gem group metrics
        result.update({'gem_group_%d_%s' % (i + 1, key): value for key, val
ue in fm.iteritems()})

        # Compute metrics over all gem groups
        result['filtered_bcs'] += fm['filtered_bcs']
        result['filtered_bcs_lb'] += fm['filtered_bcs_lb']
        result['filtered_bcs_ub'] += fm['filtered_bcs_ub']
        result['max_filtered_bcs'] += fm['max_filtered_bcs']
        result['filtered_bcs_var'] += fm['filtered_bcs_var']

    # Estimate CV based on sum of variances and means
    result['filtered_bcs_cv'] = tk_stats.robust_divide(
        np.sqrt(result['filtered_bcs_var']), fm['filtered_bcs'])

    return result
```

项目：rank-ordered-autoencoder　**作者：**paulbertens　| 项目源码 (https://github.com/paulbertens/rank-ordered-autoencoder) | 文件源码 (https://github.com/paulbertens/rank-ordered-autoencoder/tree/master/Visualizer.py)

```
def reshapeWeights(self, weights, normalize=True, modifier=None):
        # reshape the weights matrix to a grid for visualization
        n_rows = int(np.sqrt(weights.shape[1]))
        n_cols = int(np.sqrt(weights.shape[1]))
        kernel_size = int(np.sqrt(weights.shape[0]/3))
        weights_grid = np.zeros((int((np.sqrt(weights.shape[0]/3)+1)*n_rows
), int((np.sqrt(weights.shape[0]/3)+1)*n_cols), 3), dtype=np.float32)
        for i in range(weights_grid.shape[0]/(kernel_size+1)):
            for j in range(weights_grid.shape[1]/(kernel_size+1)):
                index = i * (weights_grid.shape[0]/(kernel_size+1))+j
                if not np.isclose(np.sum(weights[:, index]), 0):
                    if normalize:
                        weights_grid[i * (kernel_size + 1):i * (kernel_size
 + 1) + kernel_size, j * (kernel_size + 1):j * (kernel_size + 1) + kernel_s
ize]=\
                            (weights[:, index].reshape(kernel_size, kernel_
size, 3) - np.min(weights[:, index])) / ((np.max(weights[:, index]) - np.mi
n(weights[:, index])) + 1.e-6)
                    else:
                        weights_grid[i * (kernel_size + 1):i * (kernel_size
 + 1) + kernel_size, j * (kernel_size + 1):j * (kernel_size + 1) + kernel_s
ize] =\
                        (weights[:, index].reshape(kernel_size, kernel_size
, 3))
                    if modifier is not None:
                        weights_grid[i * (kernel_size + 1):i * (kernel_size
 + 1) + kernel_size, j * (kernel_size + 1):j * (kernel_size + 1) + kernel_s
ize] *= modifier[index]

        return weights_grid
```

项目：rank-ordered-autoencoder　**作者**：paulbertens　| 项目源码 (https://github.com/paulbertens/rank-ordered-autoencoder) | 文件源码 (https://github.com/paulbertens/rank-ordered-autoencoder/tree/master/RankOrderedAutoencoder.py)

```
def __init__(self, input_shape, output_shape):
        self.input_shape = input_shape
        self.input = np.zeros((output_shape[0], self.input_shape[0] * self.
input_shape[1] *
                               self.input_shape[2]),dtype=np.float32)
        self.output = np.zeros(output_shape, dtype=np.float32)
        self.output_raw = np.zeros_like(self.output)
        self.output_error = np.zeros_like(self.output)
        self.output_average = np.zeros(self.output.shape[1], dtype=np.float
32)
        self.weights = np.random.normal(0, np.sqrt(2.0 / (self.output.shape
[1] + self.input.shape[1])),
                                        size=(self.input.shape[1], self.out
put.shape[1])).astype(np.float32)
        self.gradient = np.zeros_like(self.weights)
        self.reconstruction = np.zeros_like(self.weights)
        self.errors = np.zeros_like(self.weights)
        self.output_ranks = np.zeros(self.output.shape[1], dtype=np.int32)
        self.learning_rate = 1
        self.norm_limit = 0.1
```

项目：Lattice-Based-Signatures　**作者：**krishnacharya　| 项目源码 (https://github.com/krishnacharya/Lattice-Based-Signatures) | 文件源码 (https://github.com/krishnacharya/Lattice-Based-Signatures/tree/master/Fiat-Shamir-based/lyu12vK.py)

```
def Verify(**kwargs):
    '''
        Verification for the signature
        i/p:
        msg: the string sent by the sender
        (z,c): vectors in Zq, the signature
        A  : numpy array, Verification Key dimension nxm
        T : the matrix AS mod q ,it is used in the Verification of the signat

    '''
    msg, z, c, A, T, sd, eta, m, k, q = kwargs['msg'], kwargs['z'], kwargs[
'c'], kwargs['A'], kwargs['T'], kwargs['sd'], kwargs['eta'], kwargs['m'], k
wargs['k'], kwargs['q']
    norm_bound = eta * sd * np.sqrt(m)
    # checks for norm of z being small and that H(Az-Tc mod q,msg) hashes t
o c
    vec = util.vector_to_Zq(np.array(np.matmul(A,z) - np.matmul(T,c)), q)
    hashedList = util.hash_to_baseb(vec, msg, 3, k)
    print hashedList, c
    if np.sqrt(z.dot(z)) <= norm_bound and np.array_equal(c, hashedList):
        return True
    else:
        return False
```

项目：wmd-relax　**作者：**src-d　| 项目源码 (https://github.com/src-d/wmd-relax) | 文件源码 (https://github.com/src-d/wmd-relax/tree/master/wmd/__init__.py)

```
def compute_similarity(self, doc1, doc2):
            """
            Calculates the similarity between two spaCy documents. Extracts t

            nBOW from them and evaluates the WMD.

            :return: The calculated similarity.
            :rtype: float.
            """
            doc1 = self._convert_document(doc1)
            doc2 = self._convert_document(doc2)
            vocabulary = {
                w: i for i, w in enumerate(sorted(set(doc1).union(doc2)))}
            w1 = self._generate_weights(doc1, vocabulary)
            w2 = self._generate_weights(doc2, vocabulary)
            evec = numpy.zeros((len(vocabulary), self.nlp.vocab.vectors_len
gth),
                               dtype=numpy.float32)
            for w, i in vocabulary.items():
                evec[i] = self.nlp.vocab[w].vector
            evec_sqr = (evec * evec).sum(axis=1)
            dists = evec_sqr - 2 * evec.dot(evec.T) + evec_sqr[:, numpy.new
axis]
            dists[dists < 0] = 0
            dists = numpy.sqrt(dists)
            return libwmdrelax.emd(w1, w2, dists)
```

项目：pybot　**作者：**spillai　| 项目源码 (https://github.com/spillai/pybot) | 文件源码 (https://github.com/spillai/pybot/tree/master/pybot/geometry/transformations.py)

```
def quaternion_matrix(quaternion):
    """Return homogeneous rotation matrix from quaternion.

    >>> R = quaternion_matrix([0.06146124, 0, 0, 0.99810947])
    >>> numpy.allclose(R, rotation_matrix(0.123, (1, 0, 0)))
    True

    """
    q = numpy.array(quaternion[:4], dtype=numpy.float64, copy=True)
    nq = numpy.dot(q, q)
    if nq < _EPS:
        return numpy.identity(4)
    q *= math.sqrt(2.0 / nq)
    q = numpy.outer(q, q)
    return numpy.array((
        (1.0-q[1, 1]-q[2, 2],     q[0, 1]-q[2, 3],     q[0, 2]+q[1, 3], 0.0
),
        (    q[0, 1]+q[2, 3], 1.0-q[0, 0]-q[2, 2],     q[1, 2]-q[0, 3], 0.0
),
        (    q[0, 2]-q[1, 3],     q[1, 2]+q[0, 3], 1.0-q[0, 0]-q[1, 1], 0.0
),
        (                0.0,                 0.0,                 0.0, 1.0
)
        ), dtype=numpy.float64)
```

项目：pybot　**作者：**spillai　| 项目源码 (https://github.com/spillai/pybot) | 文件源码 (https://github.com/spillai/pybot/tree/master/pybot/vision/recognition_utils.py)

```
def im_detect_and_describe(img, mask=None, detector='dense', descriptor='SI
FT', colorspace='gray',
                           step=4, levels=7, scale=np.sqrt(2)):
    """
    Describe image using dense sampling / specific detector-descriptor combi

    """
    detector = get_detector(detector=detector, step=step, levels=levels, sc
ale=scale)
    extractor = cv2.DescriptorExtractor_create(descriptor)

    try:
        kpts = detector.detect(img, mask=mask)
        kpts, desc = extractor.compute(img, kpts)

        if descriptor == 'SIFT':
            kpts, desc = root_sift(kpts, desc)

        pts = np.vstack([kp.pt for kp in kpts]).astype(np.int32)
        return pts, desc

    except Exception as e:
        print 'im_detect_and_describe', e
        return None, None
```

项目：sea-lion-counter　**作者**：rdinse　| 项目源码 (https://github.com/rdinse/sea-lion-counter) | 文件源码 (https://github.com/rdinse/sea-lion-counter/tree/master/data/data_preparation.py)

```
def compHistDistance(h1, h2):
  def normalize(h):
    if np.sum(h) == 0:
        return h
    else:
        return h / np.sum(h)

  def smoothstep(x, x_min=0., x_max=1., k=2.):
      m = 1. / (x_max - x_min)
      b = - m * x_min
      x = m * x + b
      return betainc(k, k, np.clip(x, 0., 1.))

  def fn(X, Y, k):
    return 4. * (1. - smoothstep(Y, 0, (1 - Y) * X + Y + .1)) \
      * np.sqrt(2 * X) * smoothstep(X, 0., 1. / k, 2) \
            + 2. * smoothstep(Y, 0, (1 - Y) * X + Y + .1) \
            * (1. - 2. * np.sqrt(2 * X) * smoothstep(X, 0., 1. / k, 2) - 0
.5)

  h1 = normalize(h1)
  h2 = normalize(h2)

  return max(0, np.sum(fn(h2, h1, len(h1))))
  # return np.sum(np.where(h2 != 0, h2 * np.log10(h2 / (h1 + 1e-10)), 0))
# KL divergence
```

项目：a-nice-mc　**作者**：ermongroup　| 项目源码 (https://github.com/ermongroup/a-nice-mc) | 文件源码 (https://github.com/ermongroup/a-nice-mc/tree/master/a_nice_mc/objectives/expression/mog6.py)

```
def __call__(self, z):
        z1 = tf.reshape(tf.slice(z, [0, 0], [-1, 1]), [-1])
        z2 = tf.reshape(tf.slice(z, [0, 1], [-1, 1]), [-1])
        v1 = tf.sqrt((z1 - 5) * (z1 - 5) + z2 * z2) * 2
        v2 = tf.sqrt((z1 + 5) * (z1 + 5) + z2 * z2) * 2
        v3 = tf.sqrt((z1 - 2.5) * (z1 - 2.5) + (z2 - 2.5 * np.sqrt(3)) * (z
2 - 2.5 * np.sqrt(3))) * 2
        v4 = tf.sqrt((z1 + 2.5) * (z1 + 2.5) + (z2 + 2.5 * np.sqrt(3)) * (z
2 + 2.5 * np.sqrt(3))) * 2
        v5 = tf.sqrt((z1 - 2.5) * (z1 - 2.5) + (z2 + 2.5 * np.sqrt(3)) * (z
2 + 2.5 * np.sqrt(3))) * 2
        v6 = tf.sqrt((z1 + 2.5) * (z1 + 2.5) + (z2 - 2.5 * np.sqrt(3)) * (z
2 - 2.5 * np.sqrt(3))) * 2
        pdf1 = tf.exp(-0.5 * v1 * v1) / tf.sqrt(2 * np.pi * 0.25)
        pdf2 = tf.exp(-0.5 * v2 * v2) / tf.sqrt(2 * np.pi * 0.25)
        pdf3 = tf.exp(-0.5 * v3 * v3) / tf.sqrt(2 * np.pi * 0.25)
        pdf4 = tf.exp(-0.5 * v4 * v4) / tf.sqrt(2 * np.pi * 0.25)
        pdf5 = tf.exp(-0.5 * v5 * v5) / tf.sqrt(2 * np.pi * 0.25)
        pdf6 = tf.exp(-0.5 * v6 * v6) / tf.sqrt(2 * np.pi * 0.25)
        return -tf.log((pdf1 + pdf2 + pdf3 + pdf4 + pdf5 + pdf6) / 6)
```

项目：simple_rl　**作者：**david-abel　| 项目源码 (https://github.com/david-abel/simple_rl) | 文件源码 (https://github.com/david-abel/simple_rl/tree/master/simple_rl/agents/bandits/LinUCBAgentClass.py)

```
def _compute_score(self, context):
        '''
        Args:
            context (list)

        Returns:
            (dict):
                K (str): action
                V (float): score
        '''

        a_inv = self.model['act_inv']
        theta = self.model['theta']

        estimated_reward = {}
        uncertainty = {}
        score_dict = {}
        max_score = 0
        for action_id in xrange(len(self.actions)):
            action_context = np.reshape(context[action_id], (-1, 1))
            estimated_reward[action_id] = float(theta[action_id].T.dot(acti
on_context))
            uncertainty[action_id] = float(self.alpha * np.sqrt(action_cont
ext.T.dot(a_inv[action_id]).dot(action_context)))
            score_dict[action_id] = estimated_reward[action_id] + uncertain
ty[action_id]

        return score_dict
```

项目：shift-detect　**作者：**paolodedios　| 项目源码 (https://github.com/paolodedios/shift-detect) | 文件源码 (https://github.com/paolodedios/shift-detect/tree/master/src/main/python/shift_detect/rulsif.py)

```
def getMedianDistanceBetweenSamples(self, sampleSet=None) :
        """
        Jaakkola's heuristic method for setting the width parameter of the Ga

        radial basis function kernel is to pick a quantile (usually the media

        the distribution of Euclidean distances between points having differe

        labels.

        Reference:
        Jaakkola, M. Diekhaus, and D. Haussler. Using the Fisher kernel metho

        remote protein homologies. In T. Lengauer, R. Schneider, P. Bork, D.

        Glasgow, H.- W. Mewes, and R. Zimmer, editors, Proceedings of the Sev

        International Conference on Intelligent Systems for Molecular Biology

        """
        numrows = sampleSet.shape[0]
        samples = sampleSet

        G = sum((samples * samples), 1)
        Q = numpy.tile(G[:, None], (1, numrows))
        R = numpy.tile(G, (numrows, 1))

        distances = Q + R - 2 * numpy.dot(samples, samples.T)
        distances = distances - numpy.tril(distances)
        distances = distances.reshape(numrows**2, 1, order="F").copy()

        return numpy.sqrt(0.5 * numpy.median(distances[distances > 0]))
```

项目：deep_architect　**作者：**negrinho　| 项目源码 (https://github.com/negrinho/deep_architect) | 文件源码 (https://github.com/negrinho/deep_architect/tree/master/darch/datasets.py)

```
def per_image_whiten(X):
    """ Subtracts the mean of each image in X and renormalizes them to unit
 norm.

    """
    num_examples, height, width, depth = X.shape

    X_flat = X.reshape((num_examples, -1))
    X_mean = X_flat.mean(axis=1)
    X_cent = X_flat - X_mean[:, None]
    X_norm = np.sqrt( np.sum( X_cent * X_cent, axis=1) )
    X_out = X_cent / X_norm[:, None]
    X_out = X_out.reshape(X.shape)

    return X_out

# Assumes the following ordering for X: (num_images, height, width, num_cha
nnels)
```

项目：deep_architect　**作者：**negrinho　| 项目源码 (https://github.com/negrinho/deep_architect) | 文件源码 (https://github.com/negrinho/deep_architect/tree/master/darch/modules.py)

```
def compile(self, in_x, train_feed, eval_feed):
        n = np.product(self.in_d)
        m, param_init_fn = [dom[i] for (dom, i) in zip(self.domains, self.c
hosen)]

        #sc = np.sqrt(6.0) / np.sqrt(m + n)
        #W = tf.Variable(tf.random_uniform([n, m], -sc, sc))
        W = tf.Variable( param_init_fn( [n, m] ) )
        b = tf.Variable(tf.zeros([m]))

        # if the number of input dimensions is larger than one, flatten the

        # input and apply the affine transformation.
        if len(self.in_d) > 1:
            in_x_flat = tf.reshape(in_x, shape=[-1, n])
            out_y = tf.add(tf.matmul(in_x_flat, W), b)
        else:
            out_y = tf.add(tf.matmul(in_x, W), b)
        return out_y

# computes the output dimension based on the padding scheme used.
# this comes from the tensorflow documentation
```

项目： pycma **作者：** CMA-ES | 项目源码 (https://github.com/CMA-ES/pycma) | 文件源码 (https://github.com/CMA-ES/pycma/tree/master/cma/sigma_adaptation.py)

```
def _update_ps(self, es):
        if not self.is_initialized:
            self.initialize(es)
        if self._ps_updated_iteration == es.countiter:
            return
        z = es.sm.transform_inverse((es.mean - es.mean_old) / es.sigma_vec.
scaling)
        # works unless a re-parametrisation has been done
        # assert Mh.vequals_approximately(z, np.dot(es.B, (1. / es.D) *
        #         np.dot(es.B.T, (es.mean - es.mean_old) / es.sigma_vec)))
        z *= es.sp.weights.mueff**0.5 / es.sigma / es.sp.cmean
        # zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
        if es.opts['CSA_clip_length_value'] is not None:
            vals = es.opts['CSA_clip_length_value']
            min_len = es.N**0.5 + vals[0] * es.N / (es.N + 2)
            max_len = es.N**0.5 + vals[1] * es.N / (es.N + 2)
            act_len = sum(z**2)**0.5
            new_len = Mh.minmax(act_len, min_len, max_len)
            if new_len != act_len:
                z *= new_len / act_len
                # z *= (es.N / sum(z**2))**0.5  # ==> sum(z**2) == es.N
                # z *= es.const.chiN / sum(z**2)**0.5
        self.ps = (1 - self.cs) * self.ps + np.sqrt(self.cs * (2 - self.cs)
) * z
        self._ps_updated_iteration = es.countiter
```

项目： pycma　**作者：** CMA-ES　| 项目源码 (https://github.com/CMA-ES/pycma) | 文件源码 (https://github.com/CMA-ES/pycma/tree/master/cma/evolution_strategy.py)

```
def result_pretty(self, number_of_runs=0, time_str=None,
                      fbestever=None):
        """pretty print result.

        Returns `result` of ``self``.

        """
        if fbestever is None:
            fbestever = self.best.f
        s = (' after %i restart' + ('s' if number_of_runs > 1 else '')) \
            % number_of_runs if number_of_runs else ''
        for k, v in self.stop().items():
            print('termination on %s=%s%s' % (k, str(v), s +
                  (' (%s)' % time_str if time_str else '')))

        print('final/bestever f-value = %e %e' % (self.best.last.f,
                                                  fbestever))
        if self.N < 9:
            print('incumbent solution: ' + str(list(self.gp.pheno(self.mean
, into_bounds=self.boundary_handler.repair))))
            print('std deviation: ' + str(list(self.sigma * self.sigma_vec.
scaling * np.sqrt(self.dC) * self.gp.scales)))
        else:
            print('incumbent solution: %s ...]' % (str(self.gp.pheno(self.m
ean, into_bounds=self.boundary_handler.repair)[:8])[:-1]))
            print('std deviations: %s ...]' % (str((self.sigma * self.sigma
_vec.scaling * np.sqrt(self.dC) * self.gp.scales)[:8])[:-1]))
        return self.result
```

项目：pycma　**作者**：CMA-ES　| 项目源码 (https://github.com/CMA-ES/pycma) | 文件源码 (https://github.com/CMA-ES/pycma/tree/master/cma/evolution_strategy.py)

```
def isotropic_mean_shift(self):
        """normalized last mean shift, under random selection N(0,I)

        distributed.

        Caveat: while it is finite and close to sqrt(n) under random
        selection, the length of the normalized mean shift under
        *systematic* selection (e.g. on a linear function) tends to
        infinity for mueff -> infty. Hence it must be used with great
        care for large mueff.
        """
        z = self.sm.transform_inverse((self.mean - self.mean_old) /
                                      self.sigma_vec.scaling)
        # works unless a re-parametrisation has been done
        # assert Mh.vequals_approximately(z, np.dot(es.B, (1. / es.D) *
        #         np.dot(es.B.T, (es.mean - es.mean_old) / es.sigma_vec)))
        z /= self.sigma * self.sp.cmean
        z *= self.sp.weights.mueff**0.5
        return z
```

项目：pycma　**作者：**CMA-ES　| 项目源码 (https://github.com/CMA-ES/pycma) | 文件源码 (https://github.com/CMA-ES/pycma/tree/master/cma/restricted_gaussian_sampler.py)

```
def __init__(self, dimension, randn=np.random.randn, debug=False):
        """pass dimension of the underlying sample space
        """
        try:
            self.N = len(dimension)
            std_vec = np.array(dimension, copy=True)
        except TypeError:
            self.N = dimension
            std_vec = np.ones(self.N)
        if self.N < 10:
            print('Warning: Not advised to use VD-CMA for dimension < 10.')
        self.randn = randn
        self.dvec = std_vec
        self.vvec = self.randn(self.N) / math.sqrt(self.N)
        self.norm_v2 = np.dot(self.vvec, self.vvec)
        self.norm_v = np.sqrt(self.norm_v2)
        self.vn = self.vvec / self.norm_v
        self.vnn = self.vn**2
        self.pc = np.zeros(self.N)
        self._debug = debug  # plot covariance matrix
```

项目：pycma　**作者：**CMA-ES　| 项目源码 (https://github.com/CMA-ES/pycma) | 文件源码 (https://github.com/CMA-ES/pycma/tree/master/cma/bbobbenchmarks.py)

```
def _evalfull(self, x):
        fadd = self.fopt
        curshape, dim = self.shape_(x)
        # it is assumed x are row vectors

        if self.lastshape != curshape:
            self.initwithsize(curshape, dim)

        # BOUNDARY HANDLING

        # TRANSFORMATION IN SEARCH SPACE
        x = x - self.arrxopt
        x = monotoneTFosc(x)
        idx = (x > 0)
        x[idx] = x[idx] ** (1 + self.arrexpo[idx] * np.sqrt(x[idx]))
        x = self.arrscales * x

        # COMPUTATION core
        ftrue = 10 * (self.dim - np.sum(np.cos(2 * np.pi * x), -1)) + np.su
m(x ** 2, -1)
        fval = self.noise(ftrue) # without noise

        # FINALIZE
        ftrue += fadd
        fval += fadd
        return fval, ftrue
```

项目：TDOSE　**作者：**kasperschmidt　| 项目源码 (https://github.com/kasperschmidt/TDOSE) | 文件源码 (https://github.com/kasperschmidt/TDOSE/tree/master/tdose_utilities.py)

```
def normalize_2D_cov_matrix(covmatrix,verbose=True):
    """
    Calculate the normalization foctor for a multivariate gaussian from it's

    However, not that gaussian returned by tu.gen_2Dgauss() is normalized for


    --- INPUT ---
    covmatrix       covariance matrix to normaliz
    verbose         Toggle verbosity

    """
    detcov  = np.linalg.det(covmatrix)
    normfac = 1.0 / (2.0 * np.pi * np.sqrt(detcov) )

    return normfac
# = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
 = = = = = = = = = = = = = = = = = =
```

项目：pyballd **作者：**Yurlungur | 项目源码 (https://github.com/Yurlungur/pyballd) | 文件源码 (https://github.com/Yurlungur/pyballd/tree/master/pyballd/domain.py)

```
def get_dXdr(self,X):
        "Derivative of compactified coordinate with respect to radial"
        L = self.L
        r_h = self.r_h
        num = ((X-1)**2)*np.sqrt((r_h*(X-1))**2 + (L*(X+1))**2)
        denom = 2*L*L*(1+X)
        dXdr = num/denom
        return dXdr
```

项目：pyballd **作者：**Yurlungur | 项目源码 (https://github.com/Yurlungur/pyballd) | 文件源码 (https://github.com/Yurlungur/pyballd/tree/master/pyballd/domain.py)

```
def get_x_from_r(self,r):
        "x = 0 when r = rh"
        r_h = self.r_h
        x = np.sqrt(r**2 - r_h**2)
        return x
```

项目：pyballd　**作者：**Yurlungur　| 项目源码 (https://github.com/Yurlungur/pyballd) | 文件源码 (https://github.com/Yurlungur/pyballd/tree/master/pyballd/domain.py)

```
def get_r_from_x(self,x):
        "x = 0 when r = rh"
        r_h = self.r_h
        r = np.sqrt(x**2 + r_h**2)
        return r
```

项目：pyballd　**作者：**Yurlungur　| 项目源码 (https://github.com/Yurlungur/pyballd) | 文件源码 (https://github.com/Yurlungur/pyballd/tree/master/pyballd/orthopoly.py)

```
def get_norm2_difference(foo,bar,xmin,xmax):
    """
    Returns sqrt(integral((foo-bar)**2)) on the interval [xmin,xmax]
    """
    out = integrator(lambda x: (foo(x)-bar(x))**2,xmin,xmax)[0]
    out /= float(xmax-xmin)
    out = np.sqrt(out)
    return out
# ======================================================================


# ======================================================================
# Nodal and Modal Details
# ======================================================================
```

项目：pyballd **作者：**Yurlungur | 项目源码 (https://github.com/Yurlungur/pyballd) | 文件源码 (https://github.com/Yurlungur/pyballd/tree/master/pyballd/orthopoly.py)

```
def norm2(self,grid_func):
        """Calculates the 2norm of grid_func"""
        factor = np.prod([(s.xmax-s.xmin) for s in self.stencils])
        integral = self.inner_product(grid_func,grid_func) / factor
        norm2 = np.sqrt(integral)
        return norm2
```

项目：GELUs **作者：**hendrycks | 项目源码 (https://github.com/hendrycks/GELUs) | 文件源码 (https://github.com/hendrycks/GELUs/tree/master/nn.py)

```
def fit(self, x):
        s = x.shape
        x = x.copy().reshape((s[0],np.prod(s[1:])))
        m = np.mean(x, axis=0)
        x -= m
        sigma = np.dot(x.T,x) / x.shape[0]
        U, S, V = linalg.svd(sigma)
        tmp = np.dot(U, np.diag(1./np.sqrt(S+self.regularization)))
        tmp2 = np.dot(U, np.diag(np.sqrt(S+self.regularization)))
        self.ZCA_mat = th.shared(np.dot(tmp, U.T).astype(th.config.floatX))
        self.inv_ZCA_mat = th.shared(np.dot(tmp2, U.T).astype(th.config.flo
atX))
        self.mean = th.shared(m.astype(th.config.floatX))
```

项目：pylspm **作者：**lseman | 项目源码 (https://github.com/lseman/pylspm) | 文件源码 (https://github.com/lseman/pylspm/tree/master/pylspm/pylspm.py)

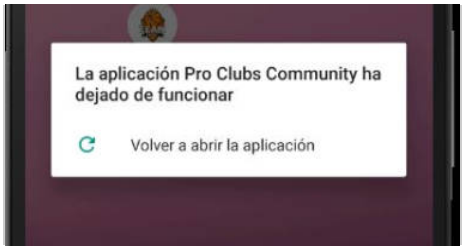
```
def normaliza(self, X):
        correction = np.sqrt((len(X) - 1) / len(X))  # std factor corretion
        mean_ = np.mean(X, 0)
        scale_ = np.std(X, 0)
        X = X - mean_
        X = X / (scale_ * correction)
        return X
```

Coding School in Europe - Codeworks™

Ad codeworks.me

מיטה יהודית במבצע נדיר

Ad ד"ר קומפורט

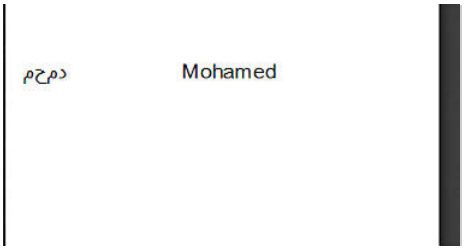


Flutter Error: java.lang.NoSuchFi...

codingdict.com

שיעור הצלחה גבוה בתרומת ביציות

Ad Dor Clinic



Convert arabic“unicode” content html or xml to pdf using...

codingdict.com

WebFTP-在线文件管理系统

codingdict.com